DECEMBER 1982 VOL. 5 NO. 4

a quarterly bulletin of the IEEE computer society technical committee on

Database Engineering

Contents

Database Engineering Bulletin is a quarterly publication of the IEEE Computer Society Technical Committee on Database Engineering. Its scope of interest includes: data structures and models, access strategies, access control techniques, database architecture, database machines, intelligent front ends, mass storage for very large databases, distributed database systems and techniques, database software design and implementation, database utilities, database security and related areas.

Contribution to the Bulletin is hereby solicited. News items, letters, technical papers, book reviews, meeting previews, summaries, case studies, etc., should be sent to the Editor. All letters to the Editor will be considered for publication unless accompanied by a request to the contrary. Technical papers are unrefereed.

Opinions expressed in contributions are those of the individual author rather than the official position of the TC on Database Engineering, the IEEE Computer Society, or organizations with which the author may be affiliated.

Membership in the Database Engineering Technical Committee is open to individuals who demonstrate willingness to actively participate in the various activities of the TC. A member of the IEEE Computer Society may join the TC as a full member. A non-member of the Computer Society may join as a participating member, with approval from at least one officer of the TC. Both a full member and a participating member of the TC is entitled to receive the quarterly bulletin of the TC free of charge, until further notice.

Membership applications and requests for back issues should be sent to IEEE Computer Society, P.O. Box 639, Silver Spring, MD 20901. Papers and comments on the technical contents of Database Engineering should be directed to any of the editors.

Letter from the Editor

This issue of Database Engineering presents the current state of "Research on Distributed Database Systems." Twelve of the leading university and industrial research groups in distributed database systems have contributed short papers that describe their current and future research efforts. The compilation of these papers provides a comprehensive survey of this important research area.

Research in distributed database systems has enjoyed a short but very active history. In the past seven years the research emphasis has been on establishing a theoretical basis for the important problems of distributed concurrency control, distributed query optimization, distributed system reliability, and design questions such as resource allocation on distributed systems. Numerous algorithms to solve these problems have been developed. As a result of this past work research on distributed database systems has attained a level of maturity in which many of the important problems are now well understood. The challenge of current research becomes one of applying this knowledge to the design of experimental research activities.

The research activities reported here demonstrate quite clearly this evolution from a theoretical emphasis to an experimental emphasis. Nearly all of the research groups represented in this special issue have constructed, or are constructing, a distributed database system upon which experiments can be performed. Performance analysis will become an increasingly important consideration in future distributed database research.

I would like to thank the contributors to this issue for their interest, enthusiasm, and willingness to meet the stringent deadlines required to produce a state-of-the-art research bulletin. My thanks also extends to my fellow editors who provided advice and assistance.

Alan R. Hevner

DISTRIBUTED DATA BASE RESEARCH AT GRENOBLE UNIVERSITY

Michel ADIBA
Laboratoire IMAG - FRANCE
BP 53X - 38041 Grenoble Cédex
Ph. (76) 54 81 45

1 - INTRODUCTION

Database research and development started at the IMAG Laboratory at the end of the sixties with Abrial's pioneer work on the network DBMS SOCRATE. Developed as a prototype this system is now commercially avaible in a great variety of computers : IBM, CII-HB, etc.

At the begining of the seventies, research activities were mainly focused on the relational model of data (11).
Data model, database design and relational database system developments were some of the first topics studied. From 76 to 79 distributed database problems were addressed mostly thru the POLYPHEME project which was part of the nation wide SIRIUS project. Several other projects started after POLYPHEME to investigate other aspects of distributed databases systems. Recent work is centered around generalized databases systems (i.e. database managing texts, images and voice) with a new project called TIGER launched in 1982. In the following we describe distributed databases activities and future research.

2 - RELATIONAL DATABASE DEVELOPMENTS (73 to 75)

This research started by analyzing database design using hierarchical and network models compared to the relational approach. A schema translator from network to relational was designed in 1974 and algorithms to obtain third normal form relational schemas were also investigated. In 1977, an experimental relational DBMS, called URANUS was implemented (12). It provided an algebraic-like language to dynamically create and manipulate a set of relations. Originally planned to provide a relational interface for the SOCRATE DBMS, it has been used as a stand-alone relational system in other projects (e.g. POLYPHEME).

3 - DISTRIBUTED DATABASE ACTIVITIES (76 to 81)

In 1976, POLYPHEME, a joint project with the CII-HB Scientific Center addressed the distributed database problem. More precisely the goal of POLYPHEME was to study how to make existing and heterogeneous databases co-operate over a general computer network (ARPA-like). In POLYPHEME we addressed the following topics :

- distributed database models : homogeneity of distributed data thru a dynamic relational model, data distribution, local and global views,
- distributed database design using a bottom-up approach,

- distributed database system architecture,
- request decomposition and optimization,
- distributed and parallel execution of decomposed queries and updates.

The work on POLYPHEME was published in several articles and theses (1,2). A distributed database system prototype was also implemented in 1979 on the french CYCLADES network.
This POLYPHEME prototype is now used by several other research groups in France to experiment on distributed database approaches.

Cooperation between distributed and heterogeneous databases is a problem which is still to be solved in its overall generality. However, we have proposed in POLYPHEME several solutions. For instance, we described in (1,6) a general relational model able to take into account the semantic aspects of existing databases. This model allows for the creation of a "global" relational view making further cooperation possible.

In order to cooperate, an existing database, which has been implemented under a network or hierarchical DBMS, must present itself as a relational automaton. This can be achieved in two different ways. First, several local application programs can be added to an existing database in order to build a relational interface. Each "relation" seen from outside the database can be manipulated through these local programs which can take advantage of local data organization and access paths. Each relational operation, i.e. get, insert, delete, update tuples, corresponds to a specific program which is activated on demand. Note that all these local programs need not be written but can be automatically generated. A second possibility has been choosen for the POLYPHEME prototype. using mechanisms developped for the URANUS project, each cooperating database is translated from its network DBMS to a relational form and only this relational copy is manipulated. Once each local database has been transformed into a relational automaton, cooperation can take place.

The definition of distributed data has been treated in POLYPHEME in the following way. The first approach to relational distributed database was to suppose that a relation was the unit of data distribution. A second step was to consider that user relations can be spread over several sites. For instance, a relation can be horizontally partitioned into several fragments stored at different sites. In POLYPHEME, we have investigated horizontal and vertical partitioning together with total and partial replications. These issues are discussed elsewhere (4) but they require a complete view mechanism which has not been implemented so far. The POLYPHEME prototype supports only the first level of distributed data and the description of the database is done by specifying for each relation the site where it is stored.

The global view is merely the union of all the local relations managed by all the sites. Note however that once the distributed database has been described the user is no longer concerned with data location, i.e. the POLYPHEME prototype provides location transparency.

On-line data definition, retrieval and modification are allowed in POLYPHEME through a simple relational language based on the one developped for URANUS. This interface is available to each POLYPHEME user when interacting with a "Global machine". Queries are expressed by combining operands, i.e. relation names and operators (JOIN, SELECT, PROJECT, ...) in a non-procedural way. Updates are expressed on individual relations.

Interpretation of a given query is done by first transforming the query into a binary tree structure (15) and then pipe-lining tuple results during tree evaluation. This tree interpretation technique has been adapted to a distributed environment taking advantage of possible parallelims between the different sites and reducing the volume of intermediate results which have to be moved from one site to another. This is done by characterizing independent sub-trees which can be evaluated locally. Several studies on these problems have been made so far. Our approach however does not require maintenance of attributes selectivity nor estimation of partial results cardinalities.

The POLYPHEME prototype is designed as a network of Abstract relational Machines. Local Machines (LM) are built around each cooperating local database in order to make their behaviour homogeneous. Global Machines (GM) allow users to interact with the distributed database, providing them with location transparency. Note that GM can store and retrieve data "locally", i.e. at the site where they are implemented. Each POLYPHEME user issues a global request (query or update) or sequences of requests considered as transactions. Global machines decompose each request into sub-requests and send them to the corresponding local machines for evaluation as described later.

Figure 1 shows the architecture of the POLYPHEME prototype as composed of a Global machine located at site A and two Local machines located at sites B and C. This is a logical architecture, physically, all combinations are possible to implement LM and GMs on real computers in a network. The prototype can easily be reconfigured to match several possibilities. In particular, we experimented POLYPHEME on the CYCLADES network with a GM located in Paris and two independent LM on the same computer in Grenoble (9). Each machine (local or global) is implemented with three basic components :

1) A relational data manager to store, retrieve local data and decompose user requests (URANUS)

2) A distributed execution monitor which provides basic communications between remote machines, remote program activation and synchronization (7,8).

3) A global (or local) execution controller reponsible for sending (or receiving) sub-trees to (from) other machines, to trigger and synchronize their parallel (or local) executions.

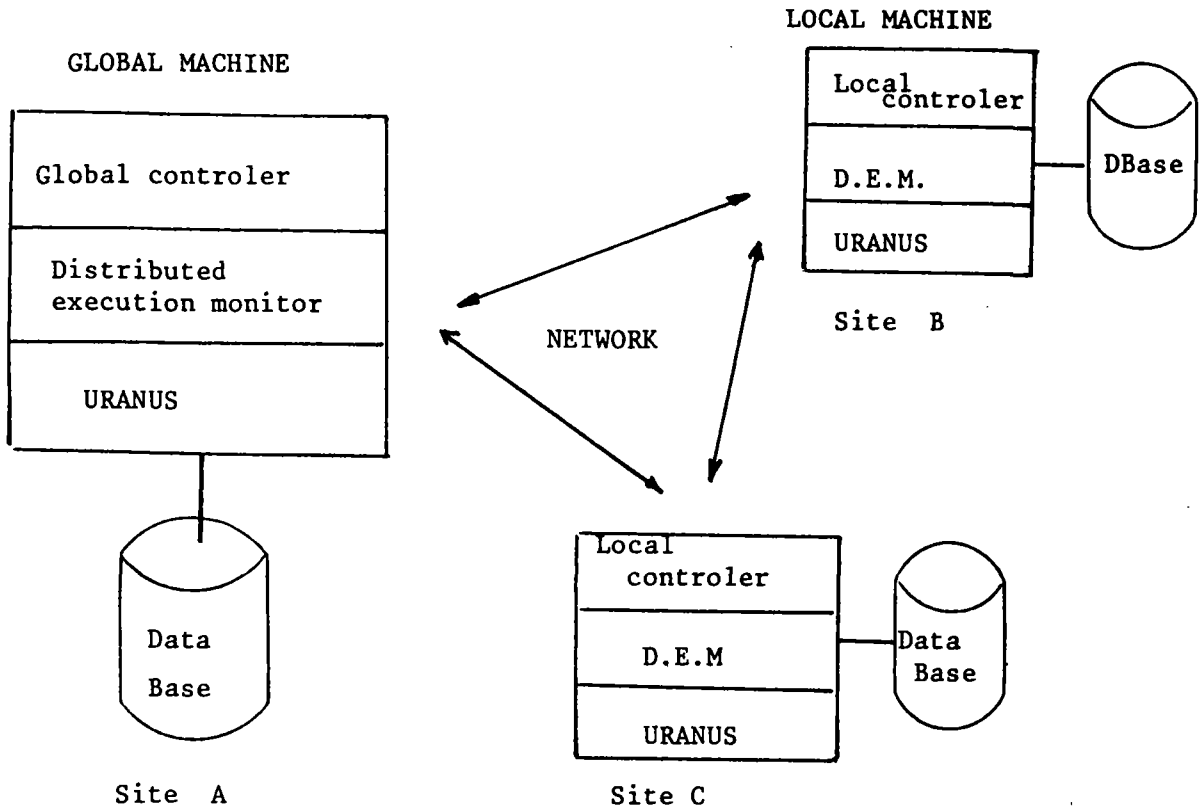


Figure 1

Following POLYPHEME, two other projects on distributed databases have been launched in 1979 and are now at an ending phase.

- MICROBE is dedicated to the design and the implementation of a relational DBMS distributed over a local network of micro-computers (12). Using Plessey micro-computers MICRO I and MICRO II (LSI-II) MICROBE can be used as a stand-alone relational DBMS which provides a MIQUEL a (micro-SQL) high level language. Several experiences have been initialized around the MICROBE system : query optimization, graphic interface (QBE-like), database integrity, etc.

- The SCOTT project done at the CII-HB Scientific Center is more concerned by the cooperation of transactions over a distributed system. Using a real banking application (electronic funds transfer) SCOTT investigates both distributed transaction design and distributed system implementation. The prototype which is currently implemented provides a distributed transaction manager with a two-phase commit protocol (17).

4 - CURRENT RESEARCH IN GENERALIZED DATABASE (> 81)

Traditionally, database systems have been used in environments where data is rather static and highly structured, namely for business applications. However, there is a growing need for systems which can manage less structured and dynamic data. Examples of such applications are computer aided design, (CAD) office automation and graphics.

Office automation and text processing are new applications which can take advantages of using database technology. Currently, capabilities of text processing machines are very limited. For instance, they do not offer very sophisticated classifications for documents or they cannot be linked easily to general databases. On the other hand, DBMS offer very poor tools for managing textual data.

Research on textual database have been made in Grenoble since 1979. For instance (19) reports on an experimental textual database system which have been built extending the network DBMS SOCRATE. One of the main result of this experimental work was to point out the inadequation of classical DBMS and to prepare a more general research on generalized database systems. After investigating several areas where databases can be used, we launched this year a new project called TIGER. In TIGER we are doing research on generalized databases in three main directions (1) Data models, (2) Systems Architecture and (3) Applications and user interface.

1) Our goal is to define a generalized data model. We are currently studing several solutions considering extensions to the relational model (e.g. ER model or RM/T), semantics networks and abstract data types.

2) We beleive that a typical generalized database system will not be an individual computer neither a centralized database. The type of systems we are looking at will be dedicated to a group from 10 to 100 people who work on alive data. Our system will include one or several database servers linked by a local network to several sophisticated working stations.

3) We are investigating several solutions to implement high level generalized interfaces which can take advantage of all the capabilities of the working station. Although graphic or form oriented interfaces will be more suitable for end-users, our system should provide a high level programming language (e.g. relational PASCAL) in order to build more general application programs.

5 - RELATIONAL MODEL THEORY

Headed by Claude DELOBEL, research in this area is essentially oriented towards the study of relational schema dependencies and their properties. It has been established the equivalence between the properties of FD and MVD with a class of Bolean expressions (21). Recent work is studying the join-project operator and associated full join or embedded join dependencies. More precisely, the goal of this work is to find a complete set of derivation rules for these kinds of dependencies (22).

6 - DATABASE GROUP

In september 1982, people who are working in the database area are M. ADIBA, A. CHAPEL, C. DELOBEL, L. FERRAT, LEE, NGUYEN GIA TOAN, J. PALAZZO, D. RIALHE and F. VELEZ. We are also co-operating with the CII-HB scientific center, particularly J.C. CHUPIN, G. BOGO, P. DECITRE, M. LOPEZ and V. JOLOBOFF.

REFERENCES

(1) M. ADIBA
"Un modèle relationnel et une architecture pour les systèmes de bases de données répartis. Application au projet POLYPHEME"
Thèse de Doctorat d'Etat, Grenoble, September 1978.

(2) M. ADIBA et al.
"POLYPHEME : an experience in distributed database system and implementation".
Distributed database, North-Holland, 1980. Proceeding of the International Symposium on Distributed Database, Paris, March 1980.

(3) M. ADIBA et al.
"A distributed database system using logical relational machines".
VLDB. Conf., Berlin, Sept. 1978.

(4) M. ADIBA, J. ANDRADE
"Expressing update consistency in a distributed database"
International Conference on Distributed Systems, Paris, April 1981.

(5) M. ADIBA et al.
"The cooperation problem between different database management systems"
IFIP TC2 Work. Conf., Nice, January 1977.

(6) M. ADIBA
"Modelling approach for distributed databases"
ECI 1978, Venice, October 1978.

(7) E. ANDRE, P. DECITRE
"On providing distributed applications programmers with control over synchronization"
Comp. Network Protocol Symp., Liege, February 1978.

(8) P. DECITRE, E. ANDRE
"POLYPHEME project : the DEM distributed execution monitor"
Int. Symp. on Distr. Database, Paris, March 1980.

(9) P. DECITRE, J. ANDRADE
"Technical outline of the POLYPHEME demonstration prototype"
Int. Symp. on Distr. Databases, Paris, March 1980.

(10) C. DELOBEL
"An overview of the relational data theory"
Invited paper, IFIP 80, Tokyo-Melbourne, October 1980.

(11) C. DELOBEL, M. ADIBA
"Databases and relational systems"
(In french) DUNOD Paris, (to appear. in 1982).

(12) NGUYEN GIA TOAN
"Distributed Query management for a local network database system"
2nd International Conference on Distributed Computing Systems, Paris April 1981.

(13) NGUYEN GIA TOAN
"URANUS : Une approche relationnelle à la cooperation de bases de données"
Thèse de 3ème Cycle, Grenoble, December 1977.

(14) NGUYEN GIA TOAN
"L'adapatiblité des bases de données par les moyens relationnels"
Congrès AFCET Informatique 78, Gif/yvette, November 1978.

(15) NGUEYN GIA TOAN
"A unified method for query decomposition and shared information updating in distributed systems"
First Int. Conf. on Distr. Comp. Syst.. Hunstville, October 1979.

(16) J. LE BIHAN et al.
"SIRIUS : a french nation wide project on distributed database"
VLDB, Montréal, October 1980.

(17) SCOT, Présentation générale, Centre de Recherche CII-HB, Rapport n° 8, Grenoble, 1980.

(18) SIRIUS :
"Actes des Journées de présentation des résultats"
Paris, Novembre 1981.

(19) I. KOWARSKI, M. LOPEZ
"The document concept in a database"
SIGMOD 82.

(20) V. QUINT, H. RICHY, X. ROUSSET, S. SASYAN, G. SERGEANT, I. VATTON
"Basic service for a local network"
International Conference on local networks and distributed office systems, London, May 1981.

(21) Y. SAGIV, C. DELOBEL, D. PARKER, R. FAGIN,
"An equivalence between relational database dependencies and a fragment of propositional logic"
To appear in JACM

(22) J. ZIDANI,
"Théorie des décompositions Généralisées"
Thèse d'Etat à soutenir (PhD. Thesis to be defended)

Distributed Database Research at the Politecnico of Milano

S. Ceri, G.Paolini, G. Pelagatti and F. A. Schreiber

Dipartimento di Elettronica, Politecnico di Milano,
P. za L. Da Vinci 32, 20133 Milano, Italy

Abstract: We briefly describe some of the on-going research in distributed databases at the Politecnico di Milano. This research is sponsored by the National Research Council of Italy, PFI, as part of the nation-wide DATANET and DATAID projects.

1. Introduction

Recent years have shown a growing interest in distributed databases, as a natural confluence of database systems and of computer networks. The development of the first prototypes has shown the practicability of the distributed database approach as well as the extreme complexity of such an approach. Many research efforts are needed both in theoretical and in practical issues in order to establish distributed databases and to provide the first commercial applications. The research project at the Politecnico of Milano does not aim to cover all the aspects of a distributed database system (like SDD-1, R* and others), but rather to give specific contributions in several areas. However, all such contributions fit within the same framework.

2. Project Framework

The Relational Model is used as a standard interface for describing the data at all the sites of the database. A Global Schema describes all the relations of the distributed database, and several Local Schemas describe the relations and fragments of relations which are stored at each site. Global relations can be horizontally partitioned into fragments according to disjoint partitioning predicates, or vertically partitioned by subdividing non-key attributes into disjoint sets, and then projecting each of them into a separate fragment; each fragment also contains the primary key. Partitioning is a major concern of the project; we assume that relations in a distributed database will be partitioned to take advantage of the possibility of locating each fragment local to the database site which mostly uses the fragment tuples (that site realistically will own access rigths and perform most of the updates on those tuples). Application programs accessing the distributed database are specified in terms of the relations of the global schema, and they don't require the notion of fragments.

3. User Language: Extended Relational Algebra

Queries against the global schema are expressed using an Extended Relational Algebra (ERA) [CePe 80b]. The major extensions provided by ERA are:
(a) the capability of partitioning a relation into horizontal fragments by means of disjoint fragmentation predicates. This feature provides the same effect produced by the GROUP BY clause of SQL, as well as more complicated horizontal fragmentations.
(b) the possibility of evaluating aggregate functions, either over the whole relation or over each fragment.

ERA is also used for the description of the mapping between relations of the Global Schema and fragments of each Local Schema (vertical fragmentation is provided by the standard projection operation). The mapping is bijective, i.e. it is possible to describe using ERA how

fragments should be aggregated in order to return the global relations. Typically, global relations are given by the union of horizontal fragments and the equijoin on the key attributes of vertical fragments. Using the inverse mapping, the user query can be transformed, in a standard way, into a "canonical" query that operates over physical fragments, and thus describes a query execution strategy. By applying algebric transformations, the canonical query can be transformed into equivalent expressions, i.e. other query execution strategies which produce the same result as the user query (see [CePe 80b]).

4. Optimization of Distributed Database Access

Optimization of query processing strategies is investigated in [PeSc 79], [CePe 82]. Query execution costs are evaluated as the cost of the required file transfers between sites. Because of the horizontal partitioning of relations, the typical operations of relational algebra now span over different fragments. The simplest way of executing them consists in collecting the fragments into global relations and to apply the operations to them, but this strategy is not the most beneficial; the alternative way consists in distributing the operations over the fragments.

The join of two or more relations which are horizontally partitioned is the most critical operation, since it requires that each horizontal fragment of any relation involved in the join be compared with all the horizontal fragments of the other one. In [CePe 80a] it is shown that the number of database sites on which the distributed join can be conveniently executed has an upper limit, which depends on the size of the involved relations. In [CePe 82] the isomorphism is shown between the problem of allocating partial join operations to database sites and the warehouse location problem. This permits the development of a linear integer program for the selection of optimal execution sites for distributed joins.

5. Flexibility of the Access Strategy

In our project, queries are compiled and distributed to all the sites participating in the execution of a multi-site query. However, the need for a certain degree of flexibility at run-time is recognized. We distinguish between:
(a) data-dependent flexibility, = the capability of modifying the query execution strategy by taking into account the value of parameters supplied by the users. We assume that the query predicates incorporate in their definitions parameters which are given a value by users at run-time (as in cursors of System R). Query execution strategies at run-time should take advantage of this additional information, that is not known at compile-time. A typical example is the simplification of a query over an horizontally partitioned relation when the query predicate implies the partitioning predicate of one of its fragments.
(b) failure-dependent flexibility, = the capability of modifying the query execution strategy because of the failure of sites at run-time. A typical example is the use of a different copy of data when the site storing the copy involved in the strategy is down.

Data-dependent flexibility is obtained using two ERA operations, called "fragment-select" and "fragment-compare". They are applied to catalogs in the Global Schema, which also have a relational description, and produce the identifiers of fragments which are actually involved in the operations. At run-time, pre-compiled database programs are activated only on the selected fragments. Failure-dependent flexibility is provided by pre-compiling alternative access programs. These features are described in [CNOPP 82].

6. Run-time System

The run-time system is designed to provide the above two types of flexibility. A master process on the activation site of the transaction coordinates the actions of slave processes at the sites participating in the execution of the transaction. The synchronization and parallelism required by the execution of the transaction is described in terms of a Control Graph (CG), executed by the master process.

The CG is similar to a Petri-net, i.e. a network of places and transitions. The local database programs which access the database at each site, the other application programs and the trasmissions of data between sites are modeled in the Petri-net as transitions. A transition can be fired by the master process when all of its input places have a token; at its completion, tokens are produced in its output places. The main differences with a Petri-net are in the following facts:

(a) transitions are not instantaneous, as they correspond to programs which require an effective execution time

(b) some special transitions can produce only a subset of the tokens in their output places. This second feature provides the intended flexibility, because by producing tokens selectively in the output places it is possible to exclude or to substitute some of the pre-compiled programs. In fact, the same effect is produced of a "cut" of part of the graph, that will not be executed. These features are described in [CNOPP 82].

7. Low Level Language: Dataflow Language

At the run-time system level, the distributed database transaction appears as a set of local programs which are executed locally to each site and of transmission programs between sites, controlled by a CG. These programs exchange data, in the form of parameters and relations storing intermediate results. Thus, it is possible to give a low-level description of a transaction using a dataflow language, which describes the interconnections between the programs. The language does not deal with the semantics of programs, but it permits the specification of the import and export session of the programs, i.e. parameters and relations exchanged. In order to provide flexibility and dynamic reconfiguration, some of the data in output from programs might be "missing"; this might produce the same effect as the "cut" in the Petri-net described above. The compilation of this low level language produces as a result the CG and the activation records for the local programs. These features are described in [COPP 82].

8. Transaction Management

Two-phase commitment is assumed as a standard technique for providing transaction atomicity. Our major concern in transaction management is to allow updates to replicated objects to proceed even in the case of failure of some of the sites. In [CNOP 82] we describe a technique, called Majority 2 Phase Locking (M-2PL), that provides this feature. According to this technique, transactions are allowed to commit if a majority of the copies in the write set is available, while updates to the other copies are propagated later by an alignment process. Thus, update transactions can proceed also in the case of single site failures or network partitioning (M-2PL assures that only one transaction at a time has the write locks on a majority of copies and can perform the update). Necessary and sufficient conditions for this approach and algorithms which satisfy such conditions are described in [CNOP 82].

9. Reliability Analysis

The need of giving figures for parameters such as system's MTBF and MTBR is strongly felt in many distributed database applications. Models and techniques for quantitatively evaluating the reliability of a distributed database system are investigated in [Schr 81], [MaRS 81], [MaPS 82]. Markov models have been used to describe each component's behaviour, and failures both dependent and independent from the state of the system have been considered.

10. Implementation

Very little effort has been devoted to the implementation of a prototype system up to now. This is partly due to the fact that we are still in the initial phase of the project, and partly to our convinction that it is better to have a good understanding of several different aspects of a system before starting its implementation. We use a network of PDP-11/34 which are in different buildings on the campus, with a star network configuration. Communication software is standard Dec-Net. We are currently implementing the system bottom-up, starting

from the run-time part; this involves the implementation of the Control Machine, to execute the CG. Our next goal is to complete the design and implementation of the low level language, That will enable us to develop simple distributed applications.

11. Distributed Database Design

An important issue is the extension of database design techniques in order to apply them to the design of distributed databases. In [CeMP 80], [CePe 80a], [CeNP 82], [CeNW 82] and [NCWD82], the problem of distributing a database schema over the sites of a distributed databases has been considered under different assumptions and stressing different aspects.

Also in the design, partitioning is a major concern of the project. [CeNP 82] deals with the specification of requirements about the horizontal partitioning of logical database objects (relations, record sets, files). The problem in the requirement specification is to determine a set of predicates which are adequate to represent all the design alternatives for horizontal partitioning; the paper introduces the notion of completeness and minimality of a set of predicates. In [CeNW 82] the result is reported of research conducted at Stanford University on the distribution of a database schema. Several candidate horizontal partitionings are considered for each database object in the schema; a mathematical model is developed for optimizing the choice of horizontal partitions, minimizing the overall transaction execution costs. The model determines the optimal solution with non-replicated fragments; replication is then introduced using an heuristic approach.

The vertical partitioning of database objects has been considered in [NCWD 82], which also reports the results of research conducted at Stanford University. Several algorithms are proposed for the vertical partition and allocation of database objects, either with or without replication.

Previous papers have dealt with distributed database design without considering partitioning. [CePe 80a] describes how to evaluate the non-additive benefits that can be obtained by replication of resources; [CeMP 80] gives a simple model for file allocation that applies to networks of minicomputers.

Our concern will be to integrate all these approaches into a single, parametric design tool that will have very general capabilities.

References

[PeSc 79] G. Pelagatti, F. A. Schreiber: Evaluation of Transmission Requirements in Distributed Database Access, Proc. ACM - SIGMOD Int. Conference, Boston, Ma., 1979.

[CeMP 80] S. Ceri, G. Martella, G. Pelagatti: Optimal File Allocation of a Distributed Database on a Network of Minicomputers, Proc. Int. Conf. on Databases, Heyden P. Co., Aberdeen, July 1980.

[CePe 80a] S. Ceri, G. Pelagatti: A Non-additive Resource Allocation Model in Distributed System Design, IEEPM Report n. 15-80, to appear on Information Processing Letters.

[CePe 80b] S. Ceri, G. Pelagatti: Correctness of Read-Only Transactions in Distributed Databases, IEEPM Report n. 16-80.

[CePe 81] S. Ceri, G. Pelagatti: An Upper Bound on the Number of Execution Nodes for A Distributed Join, Information Processing Letters, vol.12 n.1, 1980.

[MaRS 81] G. Martella, B. Ronchetti, F.A. Schreiber: Availability Evaluation in Distributed Database Systems, Performance Evaluation, vol. 1, no. 3, 1981.

[Schr 81] F. A. Schreiber: State Dependency Issues in Evaluating Distributed Database Availability, IEEPM Report n. 10-81.

[CePe 82] S. Ceri, G. Pelagatti: Allocation of Operations in Distributed Database Access, IEEE-Transactions on Computers, vol. C-32, n. 2, 1982.

[CeNP 82] S. Ceri, M. Negri, G. Pelagatti: Horizontal Data Partitioning in Data Base Design, Proc. ACM - SIGMOD Int. Conference, Orlando, Fl., 1982.

[CNOP 82] S. Ceri, M. Negri, G. Oldano, G. Pelagatti: Majority Locking in Distributed Databases, IEEPM Report n. 20-82.

[CNOPP 82] S. Ceri, M. Negri, G. Oldano, G. Paolini, G. Pelagatti: The Run-Time System of HERMES-1, IEEPM Report n. 14-82.

[COPP 82] S. Ceri, M. Negri, G. Oldano, G. Paolini, G. Pelagatti: A Dataflow Language for Distributed Database Applications, IEEPM Report n. 19-82.

[CeNW 82] S. Ceri, S. Navathe , G. Wiederhold: Distribution Design of Logical Database Schemas, revised version, IEEPM Report n. 14-81, Stanford University Report n. STAN-CS-81-884.

[NCWD 82] S. Navathe, S. Ceri, G. Wiederhold, J. Dou: Vertical Partitioning in Physical and Distribution Design of Databases, working paper, Stanford University, Stanford, CA 94305.

[MaPS 82] G. Martella, B. Pernici, F.A. Schreiber: Distributed Database Reliability Analysis and Evaluation, Proc. Second Symp. on Reliability in Distributed Software and Database Systems, Pittsburgh, July 1982.

Distributed Database Management Research at Computer Corporation of America

Arvola Chan
Daniel R. Ries

Computer Corporation of America

1. Introduction

Computer Corporation of America is strongly committed to research that advances distributed database management technology. In this paper, we summarize the major distributed system development projects in which we have recently been engaged. We also outline on-going fundamental research efforts that will enhance the capabilities of these systems.

2. SDD-1

SDD-1, originally developed at CCA, has the distinction of being the first general-purpose distributed DBMS ever developed [Rothnie80, Bernstein80a, Bernstein80b, Hammer80, Bernstein81]. SDD-1 is a distributed, relational DBMS that presents a logically centralized view of a database to its users while supporting the underlying distribution of data and processing. SDD-1 is composed of functionally identical data modules that are interconnected via a computer network and that cooperate in the processing of database queries and updates. The system permits data to be stored redundantly at many modules in order to enhance the overall reliability of the system, to improve responsiveness of the system to data access requests, to reduce communication costs in processing transactions, and to facilitate modular and homogeneous upwards scaling of database size. The Reliable Network subsystem of SDD-1 provides the system with the capability of functioning correctly despite processor and/or communications failures. A high level of survivability and robustness is achieved through redundancy of computers, communication lines, and data.

A fundamental precept of SDD-1 is that users be able to interact with the distributed system as easily as with a conventional, centralized one. Thus, the complexities associated with accessing data stored at remote sites are handled automatically by the system. Techniques have been developed for optimizing distributed queries that reduce the communication cost in processing the queries. Techniques also have been developed for updating data stored at multiple data modules in order to permit efficient execution of update transactions while ensuring that updates cannot cause database inconsistencies. Other SDD-1 pioneering innovations include the notions of transaction classes and conflict graph analysis. These notions can improve concurrency achievable in the system based on knowledge of application characteristics.

The implementation of SDD-1 was completed in 1979.

3. Multibase

Multibase is a retrieval facility for interfacing local databases that reside on different DBMSs at different nodes of a geographically distributed computer network [Smith81, Katz81, Daya182a, Daya182b, Daya182c, Landers82]. The Multibase facility will:

1. Provide high-level data definition and manipulation languages for uniformly querying all data in the multiple databases.

2. Retain local autonomy for organizing and updating databases.

3. Ensure the continued validity of all existing application programs.

Multibase includes a powerful, high-level data manipulation language for expressing global retrieval requests. Requests in this language are automatically mapped to appropriate queries over local databases. Such a mapping must be expressed in a mapping language that is sensitive to the subtle semantic qualities of the various databases and DBMSs. Because more than one local database may store information about the same real world object, the method by which a global database is derived must incorporate techniques for resolving incompatibilities among these representations. A key feature of the Multibase approach is that the same language, called DAPLEX [Shipman81], is used for both manipulation and mapping. The global schema is defined as a view over DAPLEX versions of the local databases.(1) In addition, a special integration database contributes to the definition of the global schema. This is used to record additional information (e.g. scale conversion formulas) needed to resolve differences among individual local database descriptions of the same global objects. The issue of incompatible data handling and that of "homogenization" of the heterogeneous local databases are thus treated separately. The use of DAPLEX for both view integration and view manipulation offers potential gains in the areas of data modeling and end-user query capability.

Global retrieval requests are expressed in DAPLEX without knowledge of how data is distributed among databases or which fast access paths are available for locating data in each database. A powerful optimization strategy is therefore essential for mapping global requests into efficient sequences of local operations. Multibase's global query optimization strategy is based in part on the data reduction and data movement techniques used in SDD-1 [Rothnie80, Bernstein81]. That strategy has been augmented with additional techniques designed to handle generalized objects and partially overlapping data. At the same time, the heterogeneity of database systems introduces three additional factors that affect query optimization. The different systems may vary widely in:

1. Local processing capability -- Even if the data were local, the operations required to achieve data reduction might not be supported locally.

2. Ability to operate on a block of data moved from another site -- This affects the ability to do semi-joins.

3. Speed of performing local queries -- This affects the cost estimates.

An important technique in the Multibase approach is to parameterize the different aspects of given database systems, so that they may be used as input to the optimization strategy. This avoids the expense of writing a separate optimizer for every database and / or DBMS.

Local databases may contain overlapping and sometimes inconsistent information. For example, one database may only batch update employee addresses once a month, as opposed to another database that is updated on-line. Similarly, separate local databases could contain separate, duplicate portions of a person's salary. A Multibase database administrator can specify the conditions under which overlapping information may exist, and how that data is to be handled. The DBA may specify which database to use and how data from the different databases is to be combined.

A "breadboard" implementation of Multibase was completed in early 1982. A prototype system is being implemented using Ada(2) as the implementation language, and is scheduled for completion in 1984.

---

(1) The DAPLEX versions of the local databases are straight forward representation of the relations, hierarchies, or networks that exist in the local databases.

(2) Ada is a trademark of the Department of Defense (Ada Joint Program Office).

4. DDM

The DDM is a Distributed Database Manager that is designed to be compatible with the programming language Ada [Chan81, Chan82a, Chan82b, Chan82c, Chan82d, Daya182d, Ries82]. It is a general-purpose distributed DBMS that supports the composite language ADAPLEX, which results from the integration of DAPLEX with Ada. The DDM provides transparency with respect to location, replication, concurrency, and site failure. While its functionalities are mostly patterned after those of earlier research prototypes like SDD-1 and R* [Williams81], it goes beyond these earlier systems in a number of respects. The novel features of the DDM include:

1. Support of DAPLEX. The DDM supports a core language for database definition and manipulation called DAPLEX. DAPLEX supports the notions of entities, functions, and generalization hierarchies. These high level concepts greatly reduce the complexity of handling highly structured data. At the same time, DAPLEX uses a high-level predicate syntax that makes database selection expressions easy and natural to write. The incorporation of fundamental constraints like referential integrity and subtype overlap in DAPLEX provides new challenges to efficient storage structure design and access path optimization. The use of DAPLEX also leads to the support of a very flexible scheme for data fragmentation and distribution. The DDM permits "clustering" of pairs of entities of two different types, based on the existence of a one-to-many relationship between the entities in question. The "clustered" entities are stored at the same site to ensure locality of reference. Such "clustering" often leads to "local sufficiency" [Wong81] which can greatly improve query processing efficiency.

2. Unique treatment of replicated data. The DDM distinguishes two kinds of copies of a data object: regular and backup. Only regular copies are synchronously updated; backups are updated in a batched mode in the background. Each backup serves as a "warm" standby that can readily be promoted to regular status when another regular copy fails (to retain the desired level of resiliency). Response to update transactions is improved since they can commit as soon as all the regular copies have been updated. Response to retrieval operations is also improved in the DDM due to its use of a dynamic selection strategy for the materialization of relicated data. This optimization takes into consideration the requirements of individual transactions and site availability.

3. Optimization for read-only transactions. The DDM implements a multi-version mechanism to eliminate conflicts between read-only transactions and update transactions. Read-only transactions are guaranteed to complete and are never blocked by update transactions. Likewise, no delays are ever caused by read-only transactions on update transactions.

4. More robust and efficient recovery algorithms. The DDM design solves a number of reliability problems not addressed in previous prototypes like R* and SDD-1. R* uses a two-phase commit protocol [Gray78] that is liable to block should the site that is coordinating the commitment fail. SDD-1 makes use of backup coordinators to improve resiliency. No provision, however, is made in SDD-1 to allow for recovery from the situation wherein the primary coordinator and all its backups fail simultaneously. The DDM uses an improved version of SDD-1's commit algorithm to permit automatic recovery from such "commit catastrophes". Whereas R* handles no replication, SDD-1 makes use of a spooler mechanism to collect update messages destined to a nonoperational site in order to facilitate the site's recovery. When all the spoolers for a given site fail at the same time, a "spooler catastrophe" is said to occur, and human intervention becomes necessary for recovery. The DDM makes use of the audit trails at replication sites in order to recover a failed site. It is designed to automatically recover from a "total failure" situation wherein all of the replication sites of a given data object have failed simultaneously. The DDM also employs an incremental site recovery strategy to speed up the accessibility of data on a recovering site. The portion of the data base that is stored at a site is divided into groups of logically related fragments. Each fragment group is then used as the unit for recovery. As soon as the locally stored copy of a fragment group has been rolled forward, that copy can be made accessible to new transactions immediately.

The DDM is being implemented in Ada for the purpose of transportability. The initial system is targeted for the VAX machine under the VMS operating system. It is scheduled for completion in 1984.

5. Further Research on Network Partition

General purpose distributed database systems usually restrict the operations that can take place during a network partition situation. These restrictions prevent different users from simultaneously updating different copies of replicated data. Several types of database restrictions have been proposed. One type of restriction would be to prevent all updates during communication failures. Another type of restriction would be to allow updates only in a partition that contains a "primary" copy of the data. Yet another type of restriction is to permit a transaction to perform the update only if a majority of the copies are accessible in its partition. None of the proposed or existing distributed database management systems is prepared to allow updates to copies of the same data located in different partitions of the network. Thus, to effectively use distributed databases in certain types of applications, we must either assume that communications are completely reliable, or extend the database recovery mechanisms to correctly restore mutual consistency to the copies when the partitions are reconnected. We are currently investigating two approaches that would allow updates to continue during a network partition situation, and that would restore both logical and mutual consistency after communications have been repaired.

Both approaches require the recording of auxiliary information during the lifetime of a partition situation. When communications are reestablished, the partitions exchange their respective auxiliary information. The auxiliary information, along with predefined and application specific rules, are then used to restore consistency. The approaches differ in terms of the auxiliary information that is recorded and in the ways database consistency is restored.

The first approach, called log transformations, records a history of the transactions that have been run during communication failures. After communications have been restored, histories in the different partitions are merged. Special predefined rules are used to determine the set of transactions that have to be rerun or to be run differently. These rules specify which types of transactions are commutative, which types of transactions overwrite other types of transactions, and any special type of transaction that needs to be run.

The second approach, called data patch, records initial database values when a network partition situations is detected. It also records which values have been changed during the life time of a partition. When communications are reestablished, sites in different partitions exchange their current values for data items that have been updated. These current values and their corresponding values at the instigation of the partiton are then used to update the database according to the prespecified rules. This approach would call for rules for each type of data item that can be updated. The rules specify whether to use the latest data value, apply an arithmetic function to the data values from the different partitions, or run a specific application program in order to restore mutual consistency.

The use of either of these approaches requires several additional research steps. Database administration tools are needed to specify application specific rules for each approach. Guidelines are needed to design the databases and applications that make these approaches feasible. Furthermore, performance studies are required to determine the limits of the frequencies and the durations of partition situations under which these approaches are practical.

In addition to the log transformation and data patch approaches, we are also investigating a hybrid approach that combines the two. We expect to apply the result of this research to future development of the DDM.

6. Acknowledgements

The SDD-1 project was supported by the Defense Advanced Research Projects Agency of the Department of Defense (DARPA) under contract N00039-77-C-0074. The Multibase and DDM projects are jointly supported by the Defense Advanced Research Projects Agency of the Department of Defense and by the Naval Electronics System Command (NAVELEX) under contract N00039-82-C-0226. The study on network partition is supported by the Defense Advanced Research Projects Agency of the Department of Defense and is monitored by the Air Force Systems Command at Rome Air Development Center under contract F30602-82-C-0037. The views and conclusions contained in this paper are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA, NAVELEX, or the U.S. Government.

7. References

[Bernstein80a]
Bernstein, P. A., D. W. Shipman, J. B. Rothnie, Jr., "Concurrency Control in a System for Distributed Databases (SDD-1)", ACM Transactions on Database Systems, Vol. 5, No. 1, March, 1980.

[Bernstein80b]
Bernstein, P. A., D. W. Shipman, "The Correctness of Concurrency Control Mechanisms in a System for Distributed Databases (SDD-1), ACM Transactions on Database Systems, Vol. 5, No. 1, March, 1980.

[Bernstein81]
Bernstein, P. A., N. Goodman, E. Wong, C. L. Reeve, J. B. Rothnie, Jr., "Query Processing in a System for Distributed Databases (SDD-1)", ACM Transactions on Database Systems, Vol. 6, No. 4, December, 1981.

[Chan81]
Chan, A., S. Fox, W. K. Lin, D. Ries, "The Design of an Ada Compatible Local Database Manager (LDM)" Technical Report CCA-81-09, Computer Corporation of America, November, 1981.

[Chan82a]
Chan, A., S. Fox, W. K. Lin, A. Nori, D. Ries, "The Implementation of an Integrated Concurrency Control and Recovery Scheme", ACM SIGMOD Conference Proceedings, 1982.

[Chan82b]
Chan, A., S. Danberg, S. Fox, W. K. Lin, D. Ries, "Storage and Access Structures to Support a Semantic Data Model", VLDB Conference Proceedings, 1982.

[Chan82c]
Chan, A., R. Gray, "Implementing Distributed Read-only Transactions", submitted for publication.

[Chan82d]
Chan, A., U. Dayal, S. Fox, D. Ries, "DDM: An Ada Compatible Distributed Database Manager", to appear in COMPCON Digest of Papers, 1983.

[Dayal82a]
Dayal, U., H. Y. Hwang, "View Definition and Generalization for Database Integration in Multibase -- A System for Heterogeneous Distributed Databases", Proceedings of the Berkeley Workshop on Distributed Data Management and Computer Networks, 1982.

[Dayal82b]
Dayal, U., N. Goodman, "Query Optimization for CODASYL Database Systems", ACM SIGMOD Conference Proceedings, 1982.

[Dayal82c]
Dayal, U., "Global Optimization Techniques in Multibase", Technical Report, Computer Corporation of America, in preparation.

[Dayal82d]
Dayal, U., N. Goodman, R. H. Katz, "An Extended Relational Algebra with Control over Duplicate Elimination". Proceedings of the ACM Symposium on Principles of Database Systems, 1982.

[Gray78]
Gray, J. N., "Notes on Database Operating Systems", Operating Systems: An Advanced Course, Spring-Verlag, 1978.

[Hammer80]
Hammer, M., D. Shipman, "Reliability Mechanisms for SDD-1: A System for Distributed Databases", ACM Transactions on Database Systems, Vol. 5, No. 4, December, 1980.

[Katz81]
Katz, R. H., N. Goodman, T. A. Landers, J. M. Smith, L. Yedwab, "Database Integration and Incompatible Data Handling in Multibase -- A system for Integrating Heterogeneous Distributed Databases", Technical Report CCA-81-06, Computer Corporation of America, May, 1981.

[Landers82]
Landers, T. A., R. L. Rosenberg, "An Overview of Multibase", Distributed Data Bases, H. J. Schneider (editor), North-Holland Publishing Company, 1982.

[Ries82]
Ries, D., A. Chan, U. Dayal, S. Fox, K. Lin, "Decompilation, Optimization, and Pipelining for ADAPLEX: A Procedural Database Language", Technical Report, Computer Corporation of America, in preparation.

[Rothnie80]
Rothnie, J. B., Jr., P. A. Bernstein, S. Fox, N. Goodman, M. Hammer, T. A. Landers, C. Reeve, D. W. Shipman, E. Wong, "Introduction to a System for Distributed Databases (SDD-1)", ACM Transactions on Database Systems, Vol. 5, No. 1, March, 1980.

[Shipman81]
Shipman, D., "The Functional Data Model and the Data Language DAPLEX", ACM Transactions on Database Systems, Vol. 6, No. 1, March, 1981.

[Smith81]
Smith, et. al., "Multibase -- Integrating Heterogeneous Distributed Database Systems", Proceedings of the National Computer Conference, 1981.

[Williams81]
Williams, R., D. Daniels, L. Haas, G. Lapis, B. Lindsay, P. Ng, R. Obermarck, P. Selinger, A. Walker, P. Wilms, R. Yost, "R*: An Overview of the Architecture", RJ3325, IBM Research Laboratory, San Jose, December, 1981.

[Wong81]
Wong, E., "Dynamic Re-materialization: Processing Distributed Queries Using Redundant Data", Proceedings of the Berkeley Workshop on Distributed Data Management and Computer Networks, 1981.

SURVEY OF CURRENT RESEARCH AT PRIME COMPUTER, INC. IN DISTRIBUTED DATABASE MANAGEMENT SYSTEMS

Deborah DuBourdieu

Prime Computer, Inc.
500 Old Connecticut Path
Framingham, Mass. 01701
(617) 879-2960 ext. 4015

The major focus of research into distributed database systems at Prime is an on-going investigation of support for distributed transactions [DUBO82]. The current areas of concentration are improved algorithms for multi-version distributed retrieval transactions and performance tuning.

Prime has a solid foundation for the support of distributed systems in its network, Primenet (TM), which provides complete local and remote network communication services for all Prime systems [GORD79]. Our database management system is CODASYL-compliant and provides full recovery, interactive database administration, and an interactive query language/report writer. In developing a distributed DBMS we have concentrated first upon distributed transactions.

The synchronization technique we use is two-phase locking [ESWA76] but it need be obeyed only by update transactions. For transactions which are read-only, we employ an optimization of this technique which uses multiple versions of data items. An early developer of this technology was Christopher Earnest, currently of Prime Research. Multiple versions in the context of timestamp-ordering is discussed in [BERN81], and a system whose multi-version technique is based on ours is found in [CHAN82].

The multiple version technique capitalizes on the fact that read-only transactions require simply that reads be repeatable (computationally equivalent), seeing the output of the same write operation each time the same read is performed. Each data item can be thought of as having its complete update history available (though in actuality this history is maintained only as far back as is necessary to satisfy possible requests). These previous versions are actually the same versions generated by update transactions in order to permit dynamic transaction abort after a concurrency conflict, so no extra work is done except to preserve the old images as long as there is a read-only transaction which might make use of them. Transactions receive unique monotonically increasing transaction numbers which are interpreted as version or generation numbers. When an update is performed, the new value of the data item is stamped with the transaction number of the transaction which wrote it.

When a read-only transaction begins it is given a heavily encoded

list of all transactions whose output is legal for it to read. When it performs a read, the transaction number in the data block is checked to see whether it is on the reader's list. If it is not, a chain of pointers to previous versions which begins at the data block is followed backwards in time until a version is found whose transaction number does appear on the reader's list.

A distributed transaction is implemented as a group of local transactions, one at each site the distributed transaction visits, including the originating site. The Transaction Manager at the originating site has extra responsibilities in that it must send requests against remote data files to the correct site and must coordinate events at start and end of transaction.

Locks are distributed, residing at the site with the data (we have not yet investigated management of redundant data files). This has the advantage of incurring the minimum overhead associated with acquiring locks, since it does not have the bottleneck problem and communication cost of a central lock manager site. The problem of distributed deadlock prevention is dealt with via the hybrid WOUND-WAIT algorithm [ROSE78].

This design works very efficiently for global updaters, who simply acquire a transaction number at any given local site when they first access a data file at that site. Global readers, who still use the optimization of previous images, face a new problem. They must acquire not just one list of completed transactions, but a list at every site where they access data, and all of these must be consistent. A simple but relatively inefficient method for synchronizing this activity is via a lock which must be obtained by global readers when starting and by global updaters when ending. We are investigating several more efficient alternatives, including a variation on timestamp ordering.

Synchronization of the distributed transaction when it ends is handled by a variation of the Two-Phase Commit protocol, developed by Lampson and Sturgis in [LAMP76] and by Gray in [GRAY78]. In the original statement of this protocol, an important implication is that if the coordinator fails, then the participants must wait for the coordinator's site to restart and direct the participants to the conclusion of the transaction. Using our variation, this wait is not necessary in the event that all the other participants have survived and can communicate with each other. In this case they can each exchange all the information they have about the interrupted transaction. If any of the participants received a commit, they will all decide to commit. If none heard a commit, they will all decide to abort. If not all participants survived, then the survivors will have to wait as in the original protocol. This variation improves reliability but at the expense of increased complexity in the design.

We have implemented almost the entire system, but have no

perrormance statistics yet. We have learned a great deal about the network support needed by distributed transactions. We used the network primitive of remote procedure call, with the Transaction Manager at the originating site taking the role of master, and each remote site taking the role of slave. We would now prefer to use high-level network primitives composing a complete interprocess communication service. This would facilitate the peer-to-peer communication that takes place in the (admittedly exceptional) case of coordinator failure discussed above, as well as in other error conditions.

Areas for future investigation will include management of redundant data files in the distributed environment, distributed schema management, and distributed query optimization.

REFERENCES

[BERN81] Bernstein, P. A., and Goodman, N., "Concurrency Control in Distributed Database Systems," ACM Computing Surveys, June 1981.

[CHAN82] Chan, A., Fox, S., Lin, W. K., Nori, A., and Ries, D. R., "The Implementation of An Integrated Concurrency Control and Recovery Scheme," Proceedings, Int'l. Conf. on Management of Data, June, 1982.

[DUBO82] DuBourdieu, D. J., "Implementation of Distributed Transactions," Proceedings, 6th Berkeley Workshop on Distributed Data Mgt. and Computer Networks, Feb. 1982.

[ESWA76] Eswaran, K. P., Gray, J. N., Lorie, R. A., and Traiger, I. L., "The notions of consistency and predicate locks in a database system," Communications ACM, Nov. 1976.

[GORD79] Gordon, R. L., Farr, W. W., and Levine, P., "Ringnet: A Packet Switched Local Network with Decentralized Control," Computer Networks, Vol. 3, No. 6, Dec. 1979.

[GRAY78] Gray, J. N., "Notes on database operating systems," in Operating Systems: An Advanced Course, vol. 60, Lecture Notes in Computer Science, Springer-Verlag, New York, 1978.

[LAMP76] Lampson, B., and Sturgis, H., "Crash recovery in a distributed data storage system," Tech. Rep., Computer Science Lab., Xerox Palo Alto Research Center, Palo Alto, Calif. 1976.

[ROSE78] Rosenkrantz, D. J., Stearns, R. E., and Lewis, P. M., "System level concurrency control for distributed database systems," ACM Transactions on Database Systems, June 1978.

DISTRIBUTED DATA USER'S NEEDS :

EXPERIENCE FROM SOME SIRIUS PROJECT PROTOTYPES

A.M. GLORIEUX, W. LITWIN

SIRIUS Project, INRIA, BP.105, 78153 Le Chesnay-Cédex, France

ABSTRACT

Two different approaches are presented and discussed. First the "Distributed Database Management System" approach where distributed data constitute a single logical unit for the user (prototype SIRIUS-DELTA). Then the "Multidatabase Management System" approach where distributed data constitute a collection of databases (prototypes MRDSM and MESSIDOR).

1. INTRODUCTION

SIRIUS project was set up in 1977. The goal of the project was the design of systems to manage distributed databases. SIRIUS was a, socalled, pilot project. This meant that the resources and the objectives of the project were larger than the ones of a typical research project. In particular, it was required from the project to set up research on distributed data management in French universities, to spread the knowledge of the domain within the computer industry and potential users, and to design prototype(s) of DDBMS(s) that would be sufficiently operational to be qualified for industrial use.

In order to respond to the objectives, several research studies have been set up through contracts. These studies gave rise to many theoretical results. The results have been published in virtually all important conference on databases and/or distributed systems (see the references in [LEBI80], [LITW82]).

The analysis of user's needs showed the necessity of two types of systems.

On the one hand, users need systems to manage distributed data that constitute a logically single database. This will typically be the case when data of a centralized database are distributed in order to improve performance or to provide local data processing autonomy. It would also be the case when it is desired to build the enterprise database as a database which is the integration of pre-existing databases, managed by individual data processing units that may be locally or geographically distributed. Changes to data locations obviously should not imply changes to the existing application programs.

On the other hand, users need systems to manage distributed data that constitute a collection of databases. Typically, these databases will be independently created. A collection may involve hundreds of databases that may

use different data models. Users of EURONET that interconnects more than 300 heterogeneous bibliographic databases need such a system. It is also the case of users of hundreds of databases offered by videotex systems like TELETEL (France), PRESTEL (UK) or many others under construction.

In SIRIUS, a system of the first type has been called distributed database management system (DDBMS). A system of the second type has been called multidatabase management system (MDBMS). We will present the possibilities that, in our opinion, such systems should offer to users. We will also shortly present the techniques that we implement in order to provide these possibilities. These techniques characterize SIRIUS-DELTA prototype that is a DDBMS, or MRDSM and MESSIDOR prototypes that are MDBMSs for, respectively, relational and bibliographic databases. SIRIUS-DELTA and MESSIDOR give now rise to the corresponding commercial systems. MESSIDOR is, in particular, in experimental use at INRIA library.

2. DISTRIBUTED DATABASE MANAGEMENT SYSTEMS

2.1. The possibilities

The main new possibilities that a DDBMS should provide to users are, in our opinion, the following ones :

2.1.1. The user is not aware of data distribution

The DDBMS is responsible for the identification of all data objects involved in the user's request, their localization and selection of the proper copy(ies), if any. In addition, the DDBMS should be able to handle requests from pre-existing query languages on the submitting sites.

In SIRIUS-DELTA this is achieved using first schemas associated to the DDB and related local DB's, next a common internal data manipulation language : the "pivot-language".

The "pivot-language" is a relational algebra like language that includes data selection operators (projection, restriction, join, union) and update operators (create, delete, update).

A global conceptual schema presents to users the Distributed Database as a logical single database. The Global Internal Schema contains the physical characteristics of the DDB, mainly distribution rules and mapping rules between global and local data on each local site. A local external schema is associated to the DDBMS that provides the global level with an homogeneous presentation of local data, since local DBs may be heterogeneous.

2.1.2. Various types of data distribution should be available

In order to fit the application needs, various types of data distribution and duplication must be provided and processed by the DDBMS. This is necessary if one wants to provide some DDB design facilities and processing facilities (e.g.local and parallel processing optimization, usage of preexisting local data).

A data distribution description language has been developped to describe the necessary mappings between DDB data objects and local data objects and the localization rules for these local data objects (data objects can be distributed up to an attribute value).

2.1.3. Distributed data must remain consistent

When concurrent update requests are submitted on the same or on different sites, the DDBMS must be able to keep consistent the related distributed data.

A transaction is defined in SIRIUS-DELTA as a sequence of one or several inquiries/updates enclosed by a BEG-TRANS and an END-TRANS. Distributed data consistency is achieved thanks to a distributed concurrency system that provides a unique naming of the transaction, maintains transaction atomicity and controls local accesses to data objects using locks. A 2-phase locking protocol is implemented and resources are dynamically requested in the transaction.

In case of failures, the transaction is completed or rolled-back according to its current status of completion. Strong consistency is achieved, i.e. all copies are updated or none.

2.1.4. System reliability must be achieved

In a distributed system the number of components (computers, links, etc.) increases, thus increasing the global fault tolerance.

In SIRIUS-DELTA, system availability is achieved through a dynamic reconfiguration procedure and local log files that hold distributed checkpoints. This permits hot and cold restart procedures.
In addition, failure protection is achieved during transaction commitment.

2.1.5. Heterogeneity

Heterogeneity requirements result either from a wish to offer to users flexibility on its configuration components and extensions, or from a wish to make use, as much as possible, of pre-existing components :
- data processing units that can be interconnected
- local DBMSs or data managers (DM)
- local DBs or data files.

Heterogeneity is allowed in SIRIUS-DELTA at hardware and software level. At software level, heterogeneity at DBMS/DM level is taken into account using the pivot-language. Heterogeneity at local DB or data files level is taken into account using the local external views associated to the DDB.

2.2. System architecture

In SIRIUS-DELTA DDBMS we rely on an underlying transport layer. It provides link control between two processes (message and flow control sequencing, message error-free delivery, signal message of site inaccessibility), and an adaptative routing to network topological changes.

Four basic functional layers are defined above the transport layer, that provides :
- database classical data management function ("DBMS" layer)
- distributed data handling functions (SILOE layer)
- distributed concurrency control functions (SCORE layer)
- distributed execution functions (SER layer).

Each layer involves two components :
- a global component, that provides the unified view of a distributed system
- a local component, that interacts with the global levels and the local operating system.

Cooperation between the layers SILOE, SCORE and SER is codified via the Distributed Execution Plan (DEP). A DEP is associated with each query.

All SIRIUS-DELTA layers must not necessarily be implemented upon all sites. Different configurations may exist in order to fit with specialized data processing needs and to distribute functionalities (global or local site only, consumer site, etc).

SIRIUS-DELTA is implemented upon three INTERTECHNIQUE IN2000. Its connection, through the local area network DANUBE, to HB68 with MRDS is under testing, as well as the connection to connect PHLOX, a micro-computer the DBMS of which relies upon a network data model and navigational language.

3. MULTIDATABASE MANAGEMENT SYSTEMS

3.1. The possibilities

The new possibilities that an MDBMS should offer to users are the following ones :

3.1.1. Multidatabase queries

Users should be able to formulate queries that address simultaneously data from different databases. For instance, a query to two relational databases, let it be RESTAURANTS and CINEMAS, asking for all the restaurants and cinemas on the same street. Or a query to two bibliographic databases, let it be INSPEC and PASCAL of EURONET, asking for all the documents indexed with the key word SIRIUS. All the queries should be formulated using a single language. If some databases are managed by systems using different languages, translators to the common one should be designed.

The prototype MDBMS, that we called MRDSM, allows users to formulate multidatabase queries to a collection of preexisting relational databases, managed by CII-HB commercial DBMS, called MRDS. The query language of MRDS is an extension of the MRDS language that is very close to SQL. MRDSM language allows, in particular, the users to employ database names in a query. A query may produce a relation or a set of relations. Presently, the multidatabase queries may only retrieve data, multidatabase updates are under implementation.

The prototype system MESSIDOR allows one to formulate multidatabase queries to bibliographic databases. The databases may be on different sites, or they may be all on one site (ESA, QUESTEL). They may use heterogeneous data manipulation languages (QUEST, MISTRAL). MESSIDOR provides however a single language that is very close to COMMON COMMAND SET language. The latter is recommended by European Community as the standard language for bibliographic databases. It is more than likely that COMMAND SET will become the world standard.

Presently, the difficulties related to different access procedures, language heterogeneity and the work with several databases on one-by-one basis discourage most of potential users of EURONET. Systems like MESSIDOR respond to a strong demand and should overcome the present annoying state-of-the-art.

3.1.2. Interdatabase dependencies

Users should be able to define dependencies between data that are within different databases. These dependencies may, on the one hand, relate meanings of the names, correspondance between units of measure, etc. They may, on the other hand, correspond to the interdatabase consistency constraints. In particular, they may relate replicated data.
We currently are investigating the corresponding implementation techniques, using MRDSM system.

3.1.3. Multidatabase views

Users may need a view that presents jointly data that are within different databases. In an enterprise, a multidatabase view may, for instance, present selected vital data that are gathered from the major databases of the enterprise. An MDBMS should also allow one to create a view of views or of views and databases. Finally, it should allow multiview queries.

In order to create multidatabase views, we plan to use stored collections of multidatabase queries. The corresponding investigations have just started. We plan to implement the corresponding techniques on the MRDSM system.

3.2. System architecture

MRDSM is implemented on a HB-68 computer. For MRDS, it behaves as a user, i.e. no change to MRDS software is required. For MRDSM, the MRDS system is the exclusive server of relational data and operations. If a join has to involve data from different databases, then, in order to allow MRDS to perform it, a working (temporary) database is created and filled up with the necessary data. Typically, MRDSM destroys the working databases when the multidatabase query is completed.

MESSIDOR is implemented on a MICRAL microcomputer. It is intented to be a personal front-end MDBMS. A site considers the system as a usual terminal, i.e. no site software modification is required. MESSIDOR may thus potentially be used with any database server site.

4. CONCLUSION

We have presented the possibilities that systems managing distributed data should offer to users. The basic techniques that we have developed may now be considered as validated and will satify many users. However, we continue research effort, since the gap between what we may presently offer to users and their needs still remains large.

NOTE

SIRIUS projet is supported by Agence de l'Informatique (ADI)

REFERENCES

[LEBI80] J. LE BIHAN & al. : "A french nationwide project on distributed data bases", Proc. 6th VLDB, Montreal, October 1980, 75-85

[LITW82] W. LITWIN & al. : "SIRIUS systems for distributed data management", Proc. of 2nd Int. Symp. on Distributed Data Bases, Berlin, september 1982.

R*: A RESEARCH PROJECT ON DISTRIBUTED RELATIONAL DBMS

L. M. Haas, P. G. Selinger, E. Bertino, D. Daniels, B. Lindsay,
G. Lohman, Y. Masunaga, C. Mohan, P. Ng, P. Wilms, R. Yost

IBM Research Lab
San Jose, CA 95193

Abstract

The three main goals of the R* distributed database system--ease of use, site autonomy and performance--are described. Our progress in meeting these goals, the current status of our project, and future directions of our research are also discussed.

1.0 INTRODUCTION

The experimental R* project at the IBM Research Laboratory has as its goal the development of a distributed database management system (DDBMS) meeting three key objectives: ease of use, individual site autonomy, and reasonable performance. These three objectives have different implications for a system architecture.

Local autonomy is essential in an environment where sites and communication lines may fail. To achieve resilience to failures of sites and communication lines, there can be no reliance on centralized functions or services, such as a global dictionary, or a centralized deadlock detector. Site autonomy is also essential to preserve organizational domains of responsibility in the DDBMS network. This implies that sites should perform all operations on their own data: their own binding of print names to internal names, their own decomposition of compound objects (such as relational views), and their own authorization checking, and of course, their own database accesses and updates. Site autonomy is most easily achieved in a system where databases are loosely coupled and separately managed.

As opposed to autonomy, ease of use and reasonable performance are both most easily obtained when the databases of a system are tightly coupled and centrally managed. That architecture would make it relatively simple to present the user with a single system image, making the distributed DBMS as easy to use as a single site DBMS. Better performance is also easier to achieve in a tightly coupled system where global optimization can be more easily applied. Most prototype distributed systems have a reputation for being slow. If distributed database technology is to be viable, the distributed database management system must be built with careful attention to reasonable performance for typical user operations.

How can a single DDBMS reconcile the conflicting architectural demands made on it by these three objectives? Can ease of use, autonomy and reasonable performance all be achieved by the same DDBMS? These are the questions which R* attempts to answer. The remainder of this paper gives a brief summary of the current status of R*, the progress that it has made

in answering these questions, and the directions that additional research may take in the future.

2.0 R* OVERVIEW

R* is an experimental distributed database management system being designed and implemented at IBM Research to explore issues involving relational data in a distributed environment. R* consists of a confederation of voluntarily cooperating sites, each supporting a full-function database system and communicating via messages.

R* consists of four primary subsystems. The storage subsystem is concerned with the actual storage and retrieval of data, which are represented as relatively low level objects at a single site. The data communications component provides message passing services. The transaction manager coordinates the implementation of multi-site transactions. A database language processor translates programs expressed in the SQL data definition and manipulation language [CHAM76] to operations provided by the communication and storage systems. More details can be found in [WILL82], [DANI82], [NG82].

The R* prototype is currently running on a single CPU, with several R* databases communicating within and between address spaces. Next month, R* will be installed on several machines in our laboratory. Several R* systems will run on each machine, communicating with other systems on the same and on different machines using IBM's System Network Architecture.

3.0 R* EASE OF USE

R* is an extension of the relational capabilities provided by System R to a distributed environment. Users make requests to System R in the non-procedural language SQL. As user comments indicate [CHAM80], System R was considered easy to use; hence it was logical for R* to continue using the SQL language as a user interface. This provides the user with the basis for a single site image, as the same commands are used to perform actions on both remote and local tables. Database administrators could use synonym files for table names to protect the user from any knowledge of the distributed nature of the system.

4.0 R* AUTONOMY

The next issue to be addressed is that of site autonomy. The principal technique that R* uses to achieve autonomy is decentralization. Deadlock detection, recovery, locking, catalog management, and compilation are all performed either locally or in a decentralized manner. No service is centralized. In this way, users at each site are never prevented by any other site from performing any data definition changes they wish to their own local data: revoking authorization, creating indexes, etc.

This decentralization also leads to a certain degree of resilience. For example, the high level query execution strategies produced by the query optimizer at the user's site are sent to the other sites which will do work to answer the query. At these sites, another compilation takes place

to compile that site's code for its portion of the query, consistent with the external constraints of the global plan. If some change later occurs at these subsidiary sites, they merely mark their local piece of work invalid, without disturbing any other site. When the query is next executed, that portion of the query will automatically be recompiled.

5.0 R* PERFORMANCE

We can now address the final issue in our triumvirate: performance. Compilation is a key to the performance of the System R single site relational database management system [CHAM81]. R* has also adopted compilation techniques to improve performance. In R*, query compilation is a distributed operation which involves all of the sites participating in the execution of a multi-site query. The principal issues arise in the area of catalog management, global vs. local query optimization, the level of communication between sites at both compilation and execution time, and recompilation after changes. R* has local catalogs for local objects, does remote catalog lookup as necessary, and caches the catalog information it acquires, thereby avoiding another remote lookup for subsequent queries on the same remote object.

Query optimization is performed globally by exhaustive search, using careful pruning techniques to keep the search tree from becoming too wide. Because we are convinced that simple queries can be optimized trivially, and that complex queries have a substantial local processing component in their cost, the R* query optimizer minimizes a cost which is a weighted sum of both messages and local processing [SELI80]. Once an access plan is chosen, the site which performed the query optimization (the one where the user is located) then sends out a very high level representation of the work to be done to the sites which will participate in the query execution.

Two factors affect the goal of performance at execution. The first is that most of the work to set up the query at each site--to check authorization, check semantics, choose an access path, and produce code--was done at compile time. Consequently, the only setup needed at execution time is retrieval of the compiled program at each site and verification of the user's authorization to run it. Since code was generated at compile time, each site already knows what it should do. Consequently the control messages flowing at execution time are very short: "startup", "stop", "commit", etc. The data that flows between sites to participate in the query result is blocked into large messages.

The second factor which enhances performance at execution time is parallelism. Most of the R* query processing methods, particularly the join methods, call for parallel execution at the sending and receiving sites. This means that while the sender is sending the second message-full of query (or intermediate query) answers, the receiving site is processing the first message-full. In this way, a sequence of sites can operate an "assembly line" to produce answers for a join, for example.

Any change to an object on which a query execution strategy is based (relations, access paths, authorizations) invalidates that portion of the strategy which is executed at the object's site. Some changes, such as migration of a relation, require distributed recompilation and

reoptimization of the entire query. Others, such as dropping an access path, can always be rectified by a local recompilation of the invalidated portion of the query strategy. While local recompilation is less expensive than distributed, global recompilation, it may on some occasions lead to substantial degradations in query performance. By examining the original strategy and using heuristics, R* can identify these occasions and do a distributed re-optimization of the entire query.

6.0 ONGOING RESEARCH IN R*

Having demonstrated to ourselves that R* can achieve performance, ease of use, and autonomy within the same system, our future research will concentrate in two areas: adding more function, and evaluating the technology we have invented.

We already have much of the SQL language working in a distributed environment. All the data definition statements are running, as well as insert, update, delete, and select on single tables and joins, including the use of aggregate functions. More complicated queries (such as subqueries and table migration) are expected to follow shortly.

Some relational ideas, however, have new interpretations in a distributed environment. For example, we perceive the need for opaque views that can only be decomposed by the sites which define them, rather than by any query optimizer. A new kind of object which we believe should be incorporated into R* is called a snapshot [ADIB80]. Snapshots contain a recent copy of some other object(s) which is refreshed periodically by the DDBMS. R* will also include replicated data and partitioned data. These new data types can lead to interesting conflicts between the need for site autonomy and the desire for good performance. A major objective of the R* implementation of these objects will be to reconcile these goals.

A number of commit protocols are being developed for our very general model of distributed transaction execution. These protocols are intended to minimize the number of log records written and the number of times the records are written synchronously to stable storage. Optimizations are also being introduced for read only transactions.

One of the major problems faced by the implementors of any distributed system is debugging. Test buckets which can effectively test distributed function must be developed. In addition, some tool is needed for pinpointing bugs when a test bucket fails. For the R* project, this has meant a tool which allows a user sitting at one terminal to interact with normal debugging facilities to look at any R* process running on any machine in the R* network.

Another area that is receiving considerable attention at IBM Research is high availability. A research project has been formed to study availability issues as they relate to database systems and to build a prototype of a loosely coupled local network of medium to high-end processors. A collection of R*'s running on these processors, each managing a partition of the database, will provide a single database image to the user. If a DB system fails, it may be restarted in the same or in a different processor. In the latter case, provisions exist for directly accessing the physical media from the new processor.

Now that considerable function is running in R*, we intend to explore the properties of the algorithms we have chosen to implement. For example, we will install R* on several machines in our laboratory and evaluate its performance there. The performance properties of join algorithms will also be investigated, and we may, as a result of our observations, choose to implement more join methods and discard others.

In conclusion, we are beginning a new phase of design on the R* project while continuing to evaluate the prototype we have already built.

Bibliography

Adiba, M. E. and B. G. Lindsay. "Database Snapshots", Proceedings Sixth International Conference on Very Large Databases, Montreal, Canada, October 1980, pp. 86-91 (also available as IBM Research Laboratory RJ2772, San Jose, Calif., July 1980).

Chamberlin, D. et al. "Support for Repetitive Transactions and Ad-hoc Queries in System R". ACM Transactions on Database Systems Vol. 6, No. 1, March 1981.

Chamberlin, D. "A Summary of User Experience with the SQL Data Sublanguage". IBM Research Report RJ2767, San Jose, California, March 1980.

Chamberlin, D., et al. "SEQUEL 2: A Unified Approach to Data Definition, Manipulation, and Control", IBM Journal of Research and Development. Vol. 20, No. 6, Nov. 1976, pp. 560-575.

Daniels, D. "Query Compilation in a Distributed Database System", IBM Research Laboratory RJ3423, San Jose, Calif., March 1982.

Ng, P. "Distributed Compilation and Recompilation of Database Queries" IBM Research Laboratory RJ3375 San Jose, Calif., January 1982.

Selinger, P. G. and M. Adiba, "Access Path Selection in Distributed Database Management Systems", Proceedings International Conference on Data Bases, ed. Deen and Hammersly, University of Aberdeen, July 1980, pp. 204-215 (also available as IBM Research Laboratory RJ2883, San Jose, Calif., August 1980).

Williams, R. et al., "R*: An Overview of the Architecture", Proceedings of the International Conference on Database Systems, Jerusalem, Israel, June 1982. Published in Improving Database Usability and Responsiveness, P. Scheuermann, ed. Academic Press, NY, pp.1-27.

THE DISTRIBUTED DATABASE SYSTEM VDN

Rudolf Munz

NIXDORF COMPUTER AG

Berlin, West-Germany

1. Introduction

The distributed database system VDN (a german acronym for distributed database system Nixdorf) is currently under development within Nixdorf. Its main goals are to provide an easy to use, comfortable relational database system which is able to administer data stored on different nodes of a computer network. As a distributed database system it will provide all features stated in /TRAI 82/, namely

- Location transparency: although data are geographically distributed and may move from place to place, the programmer can act as if all data were in one node.

- Replication transparency: although the data may be replicated at several nodes of the network, the programmer may treat the item as if it were stored as a single item at a single node.

- Concurrency transparency: although the system runs many transactions concurrently, to each transaction it appears as if it were the only activity in the system.

- Failure transparency: either all the actions of a transaction occur or none of them occur. Once a transaction occurs, its effects survive hardware and software failures.

VDN is not restricted to a certain type of network, that is geographically distributed or local area network. It assumes that virtual communication links exist between any pair of nodes in the network and uses a simple transport protocol. VDN does not assume a reliable network (see /HAMM 80/) but copes with all aspects of an unreliable delivery of messages.

VDN is written in ISO-Pascal. The current implementation is based on Intel 8086 microprocessors. Because of the portability of the system it can easily be moved to other processors and operating system environments.

2. VDN Objects

VDN distinguishes between five types of data objects:
records, fields, groups, subsets and links.

Records consist of a linear sequence of fields. A record type is defined by specifying a list of all fieldnames and their datatypes. For each record type there exists a primary key, consisting out of one or more record fields. Record fields can be mandatory or optional. VDN uses only two datatypes, namely NUMBER and CHAR with length attributes. For each field, value restrictions can be formulated using an interval or enumerating all legal field values.

Groups form a one level aggregation of field names and serve mainly as a shorthand for a list of record fields. Different group definitions are allowed to contain the same record field.

Subsets are specified by a query-like qualification of a subset of a record type. If a file is considered as a relational table, subset definitions allow for a horizontal partitioning of that table. Subset definitions may overlap.

Links are a shorthand for join expressions in the relational sense. That is, those joins which are know at database design point or which become important during the use of the system can be specified and named. In addition to this, it is possible to construct and maintain access paths for these static joins.

As can be seen, VDN is not a purist's implementation of the relational model. We felt free to incorporate useful features of other data models without sacrificing the advantages of the relational approach.

3. VDN Functions

The administration of the VDN system is completely freed of the physical aspects of secondary storage organization. Secondary storage is seen as a "black box". The only decisions concerning the physical database design are the construction of secondary indexes and the support of links. All other physical design decisions are handled by the system. That is, the database administration has to specify only the logical aspects of its database.

The application programmer gets the usual insert, delete and replace functions together with commands to control the begin, end or abort of a transaction. Locking (shared or exclusive) has to be stated explicitly by the application programmer. It is possible to specify that an application program does not want to wait for al lock release in case of a lock collision.

The retrieval functions are distinguished between single record access and record set access functions. Single record access is provided by a set of ISAM-like commands based on key ordered records. For record set accesses a SQL-like SELECT command is provided which creates a temporary result set. This result set can consist of fields out of different records (glued together by either static or dynamic joins) and is ordered according to the specified order criteria.

Distributed Queries, are split into subqueries using the definitions of data distribution provided by the database administration. The node where the query in entered controls the processing. Semi-joins initiated by this node are used to implement a join between different nodes.

The reason for separating single record access and record set access is that this separation is more convenient in an application program. Accessing all members of a result set is particularly clumsy if this set consists of only one member. In addition to this in many applications accesses via the primary key are dominating.

VDN users are separated into different user classes with associated privilege definitions. These privileges allow the control of all VDN commands down to the record field level.

4. Data Distribution

Subsets (horizontally partitioned relations) are the units of data distribution in VDN. Note that subset definitions are a shorthand for a qualification. The placement of a subset is a subsequent action. The subset definitions can exist without the subset being placed. This is necessary as subset definitions are allowed to overlap and need not to cover a record type. These rules are more comfortable than the fragment concept used in /ROTH 80/. The two level existence of subset definitions allow an application to use its knwoledge about data distribution without becomming dependent on it.

Redundancy can be created by overlapping subset definitions or by assigning the same subset to different nodes. The placement of subsets can be changed without affecting application programs. A subset can even be displaced without deleting the records it contains. In this case, all records of the subset are moved to the "rest" subset which exists for each record type. The "rest" subset contains all those records for which no subset placement definition applies.

Copies of data are updated simultaneously. At the end of a transaction all redundant data in the system are in the same state. There is however a mechanism to postpone the updating of copies by "disconnecting" a node from the network. In the disconnected mode, a node can read all locally available data but update only those of which no copies exist at other nodes. These redundant data will remain unaltered until "reconnect" is given which brings the local copies to the network-wide newest state. This disconnect-reconnect mechanism is useful for applications where data can be actualized on a day to day basis and an immediate actualization is not necessary. This provides some sort of site autonomy, however in another sense than used in /DANI 82/.

5. Application Philosophy

In geographically distributed networks a distributed database system supports tasks with a decentralized nature and thus a strong locality behaviour of data accesses. We assume in such an environment that 80 % of all accesses or updates concern only the local node and that the remaining 20 % involve other nodes. In VDN the price for maintaining copies increments by the same amount for each copy. In commercial environments we assume that if any copies are maintained one additional copy will be the normal case.

Distributed databases will play an interesting role in local area networks consisting of workstations and servers. The concept of a distributed database allows an easy and transparent replication of a database server for performance and reliability reasons. The actual data distribution and the locality behaviour of data accesses are of minor importance in local area networks. Distributed database systems can also provide the coupling of different local area networks via a common database. In this case, the database server implicitly acts as a gateway to another network.

Future research is concentrating on a database design aid, and user interfaces based on natural language and the integration of office information system requirements into VDN.

6. References

/DANI 82/ D. Daniels et al.:
An Introduction to Distributed Query Compilation in R*,
Distributed Data Bases (ed. H.-J. Schneider), North-Holland 1982

/HAMM 80/ M. Hammer, D. Shipman:
Reliability Mechanismus for SDD-1: A System For Distributed Databases, TODS, Vol. 5, No. 4, December 1980

/ROTH 80/ J.B. Rothnie et al.:
Introduction to a system for Distributed Databases (SDD-1), TODS, Vol. 5, No. 1, March 1980

/TRAI 82/ I.L. Traiger et al.:
Transactions and Consistency in Distributed Database Systems TODS, Vol. 7, No. 3, September 1982

ENCOMPASS

Evolution of a Distributed Database/Transaction System

John Nauman - Tandem Computers Incorporated
19333 Vallco Parkway
Cupertino, Ca. 95014
(408) 725-6000

ABSTRACT

ENCOMPASS is a database and transaction management system that provides Tandem customers high-level software to develop and install on-line transaction processing applications in a distributed environment. This paper outlines the existing parts of the ENCOMPASS product set and then discusses some of the work currently underway in each of the areas to provide further capabilities in the distributed database and transaction environments.

An Outline of ENCOMPASS

Tandem Computers developed the industry's first commercially available fault-tolerant computer system designed for on-line transaction processing [BART78]. The term "transaction", as it is used here, indicates a business action requiring interaction between an end user (bank teller or customer) and the data stored in the computer system [GRAY81]. An example of a transaction would be a bank customer's interaction with an automated teller machine. Typical applications for Tandem customers are characterized by the need to support such transactions in real time and the need to have the system tolerate faults in both the hardware and the software. Many Tandem customers make use of the system's inherent capabilities for modular expansion and distributed processing. These include two to sixteen processors connected (via a 13 million bit per second inter-processor bus) to form a node and up to 254 of these nodes connected in a long-haul network. A Tandem network features automatic route-through, reroute in case of a link failure, and best path selection. The network topology and line type/speed can be chosen by the installation to best match its requirements. The distributed aspects of the system, in both the database and transaction processing areas, are the topics of this paper.

Current Products

ENCOMPASS, the database portion of the Tandem NonStop(TM) computer

---

system software, is composed of the following individual products:

ENSCRIBE	(file system)
TMF	(transaction monitoring)
DDL	(Data Definition Language)
ENFORM	(query/report processing)
PATHWAY	(application environment)
ENABLE	(application generation)
TRANSFER	(delivery system)

ENSCRIBE provides access to the data stored in the database. Data can be stored in any of several storage organizations. The data and associated indexes can be physically distributed across any of the processors within a single node or across geographically distributed nodes. Application programs can access the data without regard for its geographical location. ENSCRIBE performs all processing necessary to locate the data within the Tandem network, however, since only one copy of the data actually exists, some network delay may be experienced if data is located at other nodes of the network. The data model supported by ENSCRIBE is the relational data model. The lack of embedded structure in the relational model is critical to the system's distributed capabilities, allowing the data to be accessed and moved within the network with relative ease.

The Transaction Processing Facility (TMF) supports the database aspects of the system by providing reliable transaction processing for applications running within the system [BORR81]. This processing can include access to one or more databases on a single node or distributed throughout a network. For transactions processing both local and distributed data, TMF guarantees the database modifications of an application program will occur consistently -- that is, modifications will either occur in all databases and nodes affected or none of them. This is done with a two-phase commit protocol for transactions. The installation may choose this transaction consistency by specifying that TMF is to control access to the database files. TMF is integrated with Tandem's languages and utilities to allow the installation to easily make use of the capabilities.

Data Definition Language (DDL) is a passive data dictionary that allows the installation to describe its databases, whether local or distributed. The dictionary which describes the databases may reside at a single node or may be replicated at multiple nodes of network. DDL also aids in the generation of the underlying ENSCRIBE files, and produces record declarations, that Tandem's programming languages can use. In addition, Tandem's high level database products access the data dictionary directly.

ENFORM is a query and report product that offers a relational interface for specifying queries about data stored in one or more databases. Since ENFORM access to files is via the ENSCRIBE interfaces and, since the descriptions of the records come from the data dictionary, ENFORM can support queries or reports about data physically distributed across the network. The typical ENFORM query

might involve joining of files from different nodes of the network, and then selecting and projecting the desired information from the relation thus formed. The binding of the query to the data itself is done during the processing of the query by utilizing information in the data dictionary. Because of this binding method, data may be moved about in the system with minimal impact to existing installation queries or reports. ENFORM consists of two distinct processes -- a front end compiler and report writer and the back end query processor. The front end contains the logic to compile the queries and to form the requested reports. The back end contains the heuristic techniques that perform strategy selection, optimization, and access the pertinent relations. Because these processes can run separately, query processing itself can be distributed to different network nodes.

PATHWAY provides the system's environment for distributing transactions. PATHWAY separates the application into requestors (which handle terminals and are coded in a language called Screen COBOL) and servers (which provide the database capabilities). Servers can be written in any of the TANDEM supplied languages, but are most typically written in COBOL. The requestors and servers, once written, can run in different CPUs and even in different nodes of the network with no change required to the application itself. The ability to geographically distribute the application processing in conjunction with the reliable transaction and database aspects of TMF, gives the installation an additional level of flexibility when deciding how best to distribute the work.

ENABLE is an application generator driven almost entirely from the DDL description of the database (local or distributed). ENABLE generates, from this DDL description, a PATHWAY application to allow simple processing of the database. ENABLE thus shields the installation from almost all aspects of application programming, save the description of the database itself, while still providing the benefit of applications operating on distributed databases in a TMF and PATHWAY environment. While most of the application work on Tandem systems is done by the programmers writing in the PATHWAY and TMF environments, ENABLE is being used to generate an increasing number of applications, either as models or as finished applications.

TRANSFER provides reliable, time-staged delivery of information. TRANSFER applications can send information anywhere within a Tandem network and be guaranteed that the information will be delivered whether the addressee is currently active on the system or not. TRANSFER offers a generalized capability to send radically different kinds of data. For example, a single TRANSFER application can send ASCII data, database records and dot matrix representations in the same package, thus more effectively providing the information required for a business transaction. TRANSFER uses the facilities of TMF to ensure delivery of information it handles and PATHWAY to provide an easy-to-use interface.

Future Directions

From the above, it can be seen that ENCOMPASS addresses many of the classical problems in distributed database processing. Even so, many interesting problems remain to be addressed. Tandem is identifying those features absent from the existing system which would further satisfy the goals of fault tolerance, online transaction processing, modular expansion, and distributed database and transaction processing. The remainder of this paper discusses some of the aspects of ENCOMPASS that are under investigation.

Distributed databases in Tandem systems today use network connections which typically operate at 56KB. TMF processing differs between the "long haul" networks connected in this way and the very fast local links between processors within a node. In the initial TMF design, the number of transactions operating in a distributed environment was assumed to be self-limited by the network capabilities. Now, however, Tandem has announced hardware and software support for the connection of up to 14 systems (up to 224 processors) in a very fast local network. This necessitates a change in the original TMF assumptions. Because of the local network speed, transactions are more likely to be distributed. Even so, the number of processors actually involved in the transaction will probably not approach the total number in the system. We see the need for a more sophisticated mechanism for tracking which processors have done work on behalf of a transaction. In this way, we can ensure processors in the local network aren't required to deal with "uninteresting" transaction activity [BORR81].

A second area which we are beginning to investigate is replicated data in a distributed database environment. Currently, ENSCRIBE and TMF do not directly support the replication of data at different nodes of the network. Some internal Tandem applications have implemented database replication by making use of ENSCRIBE and TMF. Current requirements seem to be that data is locally consistent and updated periodically (rather than instantly) as a result of activity at other nodes. The internal applications solve the problem of data contention for the replicated databases by assigning records within the database a "home node" and allowing update only by that node. Changes are then periodically propagated to the other nodes. This has proven successful so far and, coupled with the ability to run transactions spanning nodes, does not seem to overly restrict the application's capability. Although the "home node" and deferred update concept seems adequate for the applications we have investigated, we continue to investigate other approaches suggested in the literature.

As distributed databases and transactions become more prevalent, the need for additional participation and control by the data dictionary increases. We are currently investigating modifications to the existing DDL product to allow it to provide active multi-node services with respect to databases and the transactions. We are investigating several items in the database dictionary area that should further enhance the system's database and transaction processing in a distributed environment. The first of these is the ability to dynamically relate dictionaries at various distributed nodes to one

another. This will allow the installation to describe the database at the location that it is used, combine the descriptions from other nodes of the network, and use this combination to interact with the data. In addition, research is being done into how to produce an "active" data dictionary in a distributed environment. This centers on the ability to relate objects within a distributed environment and to permit or disallow actions based on these relations. Finally, we are investigating modifications to the current distributed security scheme to enhance the protection offered in a distributed environment. The investigations involve "hardening" the network against possible breach if a single node of the network is taken over. Such security enhancements are a requirement for installations that run their businesses based on distributed transaction processing.

In PATHWAY we are investigating the ability to control the distributed transaction environment in which PATHWAY applications run. We feel that the ability to exercise central control over distributed applications is an important capability. This central control, and the capability for PATHWAY applications to be knowledgable about the network, are particularly promising areas of investigation.

Finally, in the TRANSFER product, we are looking at various ways that Tandem and its customers can utilize the reliable, time-staged delivery mechanisms provided by TRANSFER. The current TRANSFER and TRANSFER/MAIL products will be followed by a TRANSFER/FAX package, which will allow an installation to make cost effective use of its Tandem systems to do a significant amount of its data and information distribution. Further, TRANSFER is being investigated as a basis for additional functions, such as stored voice, voice synthesis and ASCII/FAX conversion. Our investigations indicate that TRANSFER can play a central role in the management of a company's distributed data and information resources.

Summary

The above clearly indicates that, although ENCOMPASS provides the groundwork for distributed database and transaction processing, there are still several areas in which it can be enhanced to satisfy the growing requirements for distributed on-line transaction processing.

Bibliography

BART78, J. Bartlett, "A NonStop Operating System," Eleventh Hawaii Conference on System Sciences, Jan. 1978.

BORR81, A. Borr, "Transaction Monitoring in ENCOMPASS," Proceedings of the Seventh International Conference on Very Large Databases, Sept. 1981.

GRAY81, J. Gray, "The Transaction Concept: Virtues and Limitations," Proceedings of the Seventh International Conference on Very Large Databases, Sept. 1981

The DDBS POREL: Current Research Issues and Activities

E.J. Neuhold
B. Walter

University of Stuttgart
Azenbergstr. 12
D-7000 Stuttgart 1
Fed. Rep. of Germany

Abstract: We briefly describe current research issues as well as current activities related to POREL, a distributed database system designed and implemented at the University of Stuttgart.

1. Introduction

The DDBS POREL /NEUH82/ was designed and implemented in the years 1977 - 1982 at the University of Stuttgart. The implementation of a first prototype has just been finished. Based on the experience of this implementation further research studies have been made and are partially still in progress. In the following chapters each of our current subprojects related to distributed database systems will be briefly surveyed, then some of our future plans related to the DDBS POREL will be sketched.

2. Surveillance of Remote Sites

The processing of a distributed transaction is managed by its site-of-origin. Assume that a transaction T has been initiated at site A and that some part of T has to be processed at some remote site B. Assume further, that A is waiting for a response message from B. Now, if B crashes or becomes unavailable due to communication breakdowns, two different strategies are possible:
- A waits until the awaited message arrives. However, in the worst case A must wait forever.
- A backs out the affected transaction.

The second strategy requires some mechanisms for detecting the unavailability of remote sites. One such mechanism is the newly developped RSC-protocol (RSC = Reliable Surveillance in Computer-networks) /WALT82a/ which has the following characteristics:
- Maintenance of a table at each site of the network showing for all remote sites whether they are currently available or not.
- Synchronization of the clocks in the system.
- Robustness against any number of site crashes and communication breakdowns including network partitioning.
- Distributed Control.
- Minimality in the given context, i.e. a minimum number of messages is needed to perform the surveillance task and a minimum of time is needed to propagate detected crashes and recoveries.

Among others the RSC-protocol supports WATCH-primitives. If such a WATCH-primitive is called for a remote site B, then a signal is passed to the calling module as soon as B becomes unavailable (available). A desciption of recovery strategies for distributed transactions using WATCH-primitives can be found in /WALT82a/ and in /WALT82b/.

3. Deferred Updating of Secondary Copies

In distributed systems it is not so easy to implement read-only transactions with short response times and low processing costs which nevertheless provide a consistent view of the database. However, the observation that in many applications most of the users do not really need data at the highest level of actualization led to a new solution of this problem /WALT82c/.

The data in our system may be stored partially redundant. For each object in the database there is one primary copy and a set of secondary copies. The primary copies of different objects may be located at different sites of the network. At each site either one secondary copy of an object may be located or none /NEUH82/. Each copy is stored twice on secondary storage, i.e. for each page there are two physical representations.

In this context an updating protocol has been developed, which immediately updates the primary copy of a data object whereas the updating of the secondary copies is deferred to some later point of time. The primary updates as well as the deferred updates possess the atomic property, hence, each update transaction has two commit points. From the user's point of view an update transaction terminates after having committed all its primary updates. So-called read-only-primary transactions provide the user with the most actualized view of the database at normal costs and read-only-secondary transactions provide the user with an earlier but nevertheless consistent view of the database at low costs. However, in periods with only a small amount of updates this 'earlier' view also may reach the highest level of actualization.

The deferred updates are processed making use of stable storage. At first only one of a copy's two physical representations is updated whereas the other representation is used for the processing of read-only-secondary transactions. After the deferred updates have been committed, new arriving read-only-secondary transactions are now processed using the 'new' representation. The 'old' representation can be updated as soon as all transactions using this representation have terminated. It is this newly developped strategy for switching from the 'old' to the 'new' representation which enables read-only-secondary transactions to be processed practically without synchronization and without any delays. In systems with only a small amount of updates this strategy would be suitable for updating primary copies as well. Furthermore, the proposed algorithm supports local load balancing strategies.

4. DDBS and Open Systems

ISO's Open Systems Interconnection Architecture was developed in order to standardize network protocols. The highest layer of this architecture is the so-called application layer (layer 7). In the case of distributed database systems this layer would include the database operating system. An overall architecture of such an application layer is being developed to meet the requirements of distributed database systems.

The investigation includes a comparison of the various approaches to distributed data management to identify common primitives used in the network sensitive parts of a DDBS. Such commonly used primitives which should be contained in an adequate layer 7 design include among others:

- Routines for the manipulation of tables, which are used to contain the current processing states of transactions.
- Mechanisms for execution control such as the above mentioned WATCH-primitives.
- Protocols for the reliable transfer of data and control.

In this context a model for transaction-oriented communication in distributed systems has been developped /ROTH82/. This model identifies a set of primitives to support typical communication patterns used between the various modules which have to cooperate in order to process distributed transactions.

5. Formal Techniques

The application and evolution of formal techniques have always been an important activity in our research group. For instance, the Vienna Development Method (VDM) has been used for the specification of a 3 level external/conceptual/internal schema view of a relational database system /NEUH81/ and for specifying parts of an application development system /STUD82b/. Currently formal techniques are used in the context of distributed database protocols. Since such protocols (e.g. for synchronization or for recovery) are often rather complex, it is not obvious whether they are correct or not. Therefore, formal techniques are required for specifying the protocols and for giving rigorous proofs of their characteristics.

Higher level petri nets are used as a base for the development of so-called Timed Predicate Transition Nets (TPrT-Nets) /WALT82b, WALT82d/. TPrT-Nets include the possibility to model minimum and maximum message delays as well as timers; they have been used to model the above mentioned RSC-protocol and to verify its properties 'freedom from deadlocks' and 'bounded'. Freedom from deadlocks means that under each reachable marking at least one transition is enabled. To prove this property for a TPrT-net one has to show that this property holds for the untimed net and that the availability periods of the tokens which enable some

transition do overlap. Examples have been given to show that the timing of places is superior to the timing of transitions. The major advantage of TPrT-Nets is that they are suitable for symmetric protocols with an arbitrary number of participants.

Another formal technique for specifying protocols in distributed systems is the Behavioural Description Language (BDL) which is currently under development /KARJ82/. BDL is based on an event-oriented model, in which interactions and sequences thereof are fundamental concepts. Interactions between processes represent synchronized communication over explicit interaction points. Processes which may have concurrent and nondeterministic behaviour are denoted by so-called behaviour expressions. An algebra has been defined for the operations of sequential, conditional, and parallel composition of processes and for the restriction of interaction points. Out of a behaviour expression a linear discrete sequence of interactions can be derived, hence temporal logic can be used for proving properties of BDL-specified systems.

6. User Interface Related Activities

To improve the usability of (distributed) database systems the so-called Application Development and Support System (ADS) has been developed /STUD80, STUD82a/. ADS, which is designed to be implemented on top of the language interfaces of POREL offers dialogue supported facilities for selecting, combining, and executing prepared application programs from an application library; it can be used in the context of any relational database system.

Another subproject is oriented towards extending the POREL-architecture to a three-level-architecture. This includes activities for developping dialogue interfaces for
- defining a conceptual schema using the THM semantic data model /SCHI82/,
- deriving a relational schema from the conceptual schema,
- defining subschemas which are mapped to the conceptual schema,
- defining method skeletons and specifying the appropriate input data at the conceptual schema level.

It should be noted that the latter activity differs from the above mentioned ADS in so far as it is constructed on top of the conceptual schema level whereas ADS directly uses the relational interface.

Finally in a new subproject an interactive tool will be developed which enables the user to determine a suitable distribution as well as suitable replication of data.

7. Future Plans

Concerning the DDBS POREL the following activities have been planned:
- Extensive testing of the prototype in order to improve the per-

formance and the reliability of the current implementation.
- Evolution of the current transaction management facilities to include protocols of higher reliability.
- Investigation of some problems which are related to the execution of host language initiated transactions, e.g. how to avoid a permanent 'jumping' of the cursor from one site to another when a relation is accessed following some logical order which does not correspond to its physical ordering.

Further activities will be related to the portability of POREL in order to enable an installation of the system on other computers.

8. References

/KARJ82/ Karjoth, G., "A Behavioural Description Language for the Formal Treatment of Protocols in Distributed Systems", in preparation, University of Stuttgart, 1982.

/NEUH81/ Neuhold, E.J., T. Olnhoff, "Building Data Base Management Systems through Formal Specification", Proc. Int. Coll. on Formalization of Programming Concepts, Lecture Notes in Computer Science 107, Springer Verlag, 1982.

/NEUH82/ Neuhold, E.J., B. Walter, "An Overview of the Architecture of the Distributed Data Base System POREL", in: H.J. Schneider (ed.), "Distributed Databases", North Holland Publishing Company, 1982.

/ROTH82/ Rothermel, K., "Primitives for Transaction-Oriented Communication Systems", in preparation, University of Stuttgart, 1982.

/SCHI82/ Schiel, U., "The Temporal-Hierarchic Data Model (THM)", TR 10/82, University of Stuttgart, July 1982.

/STUD80/ Studer, R. "A Dialogue Interface for Database Applications", Proc. Very Large Databases 6, Montreal, 1980.

/STUD82a/ Studer, R., "Concepts for the Interactive Development and Usage of Application Systems", Dissertation, University of Stuttgart, March 1982 (in German).

/STUD82b/ Studer, R., "Using VDM for the Development of Interactive Application Systems", in: Y. Ohno (ed.), "Proc. Int. Symp. on Current Issues of Requirements Engineering Environments", North Holland Publishing Company, 1982.

/WALT82a/ Walter, B., "A Robust and Efficient Protocol for Checking the Availability of Remote Sites", Computer Networks 6:3, July 1982.

/WALT82b/ Walter, B., "Transaction Oriented Recovery Concepts for Distributed Data Base Systems", Dissertation, University of Stuttgart, June 1982 (in German).

/WALT82c/ Walter, B., "Using Redundancy for Implementing Low-Cost Read-Only Transactions in a Distributed Database System", Working Paper, University of Stuttgart, August 1982.

/WALT82d/ Walter, B., "Timed Predicate Transition Nets: A Tool for Modelling and Analyzing Protocols in Distributed Systems", Working Paper, University of Stuttgart, October 1982.

A Structural View of Honeywell's Distributed Database Testbed System: DDTS

By

Said K. Rahimi
Mark D. Spinrad
James A. Larson

Honeywell Corporate Computer Sciences Center
10701 Lyndale Avenue South
Minneapolis Minnesota 55420
(612)887-4570

1. INTRODUCTION

Distributed database management systems have raised new implementation questions, in addition to those usually asked for the conventional centralized systems. One way to answer these questions is by conducting experiments using a distributed database testbed system. This paper presents the initial design of one such testbed in Honeywell, called the Distributed Database Testbed System (DDTS).

DDTS became operational at Honeywell's Corporate Computer Sciences Center (CCSC) in July, 1982. DDTS is a testbed in that the system can be used for experimental evaluation of different issues in distributed database management systems. The first version of DDTS is an integrated, rather than federated distributed database management system in the sense that each node contains the same conceptual schema that describes all of the data in the system.

Section 2 describes the basic architecture of DDTS, and section 3 describes some of the experiments planned for using this testbed.

2. DDTS ARCHITECTURE

The ANSI/SPARC 3-schema architecture was generalized to a 6-schema architecture (Figure 1), in order to provide flexible user interfaces in a distributed system. This *DDTS information architecture* consists of the following six inter-related levels of data description:

USER SCHEMA, a description of a user's view of the database, specified in terms of any of several data models. One such schema may exist for each user group. User schemas are not supported in the current version of DDTS.

ECR SCHEMA, a description of a portion of the database to be accessed by a user or user group, specified in terms of the same data model as the global conceptual schema. ECR, the Entity-Category-Relationship data model [WEEL80] is a generalization of Chen's Entity-Relationship (ER) data model [CHEN76]. In the current version of DDTS, ECR schemas are not supported.

GLOBAL CONCEPTUAL SCHEMA, a semantic description of the total database content. The global conceptual schema is specified using the ECR data model. Users use a high-level language, called GORDAS (Graph ORiented DAta Selection language) [ELMA81], as a query and update language. Currently, users may specify GORDAS commands directly against the global conceptual schema. The data definition language for the ECR data model includes the capabilities to specify general constraints such as attribute domains. Update transactions are specified without integrity constraints, and the transaction compiler is designed to automatically modify the transaction to check for semantic constraints defined in the global conceptual schema [ELMA80]. The transaction will be examined as a whole, so that only necessary checks are generated. Automatic integrity constraint check generation has not been implemented in the initial phase of DDTS.

GLOBAL REPRESENTATIONAL SCHEMA, a syntactic representation of the total database. The global representational schema is specified using the relational data model. User requests, stated in GORDAS, are automatically translated to the relational operators of the representational schema.

LOCAL REPRESENTATIONAL SCHEMA, a syntactic representation of the data resident at a specific node. The local representational schema is specified using the relational data model. Transactions against the global representational schema are transformed into several local transactions against selected local representational schemas. Transaction optimization is accomplished by using data flow analysis to maximize inter-command parallelism [HEVN81].

LOCAL INTERNAL SCHEMA, a local representation of data resident at a specific node. The local internal schema is specified using the data model of the host database management system. Local transactions are translated to sequences of commands against the local internal schema. Currently all local internal schemas are IDS/II CODASYL network schemas [HONE72].

The DDTS system architecture consists of three types of abstract processors (Figure 1):

INTERFACE PROCESSOR (IP) accepts transactions from the user expressed in terms of a user schema, and modifies those transactions to be expressed in terms of the global conceptual schema. The interface processor removes all of the differences in style used by different users to express their transactions, and produces a common style of transaction expression convenient for the transaction processor. Interface processors are not supported in the current version of DDTS.

TRANSACTION PROCESSOR (TP) modifies a GORDAS transaction to check for the semantic constraints defined in the global conceptual schema. At compile-time, TPs compile transactions and develop strategies for processing transactions by appropriate data processors. For interactive transactions execution is followed by compilation. Execution performance can be gained by storing a compiled transaction

at local nodes. A compiled transaction can be executed many times without recompilation.

DATA PROCESSOR (DP) accepts commands expressed using the local representational schema and executes those commands against the portion of the database for which it is responsible.

The data, transaction, and interface processors are designed as "Guardians" [LISK79], abstract processors which support multiple processes with shared memory. Processes in different guardians communicate through messages sent to, and received from, ports (one-way message queues).

Reliable execution is accomplished as follows. A master/slave relationship between a distributed execution monitor (DEM) in a transaction processor, and local execution monitors (LEM's) in selected data processors, is established to carry out the parallel execution and two-phase commitment of each transaction. The initial version of DDTS requires all data replicas (if any) to be available for updates to occur. Much of the recovery and restart capabilities in DDTS are built upon facilities existing in the local DBMSs. These include write-ahead logs and transaction backout and restart.

Concurrency control uses the locking facilities and conflict detection mechanism of IDS/II to build a distributed deadlock prevention technique. In this technique, locks are granted "conditionally" on the basis of transaction timestamps (age) that prevent deadlocks [ROSE78]. When a transaction is selected to be restarted, the DEM is responsible for restarting the transaction at all of the related data processors.

3. STATUS AND FUTURE PLANS

Currently, DDTS is functional in a multicomputer environment at Honeywell's Corporate Computer Sciences Center (CCSC). Three Honeywell Level 6 minicomputers form the initial DDTS system. Approximately 20,000 lines of C-language code comprise the DDTS software resident on the Level 6's. Capabilities of the initial system include:

- translation of queries from GORDAS to a canonical representational form;
- site selection of data to be used for transaction execution, using a data clustering algorithm;
- distribution and synchronization of transaction compilation and execution; and,
- translation from the representational form to the network data model (IDS/II).

Implementation of stored transaction capabilities and GORDAS updates is scheduled for completion in 1982. A reliable transaction

processing scheme, based on a quorum-based commit protocol [SKEE82] and distributed concurrency control algorithm, is also scheduled for completion in 1982.

The follow-on version of DDTS and our future experimentations plan call for:

- database design experiments to determine tradeoffs involving data distribution, data replication, and tools to support distributed database design;
- distributed data dictionary facilities;
- evaluation and enhancements of materialization and access planning;
- evaluation of several different concurrency control algorithms.

REFERENCES:

[CHEN76] Chen, P., "The Entity_Relationship Model: Towards a Unified View of Data," ACM Transactions on Database Systems, vol. 1, No. 1, March 1976.

[ELMA80] Elmasri, R., "Semantic Integrity in DDTS (Distributed Database Testbed System)," Tech. Rep. HR-80-274, Honeywell CCSC, Bloomington, Minnesota, December 1980.

[ELMA81] Elmasri, R., "GORDAS: A Data Definition, Query and Update Language for the Entity-Category-Relationship Model of Data," Tech. Rep. HR-81-250 Honeywell CCSC, Bloomington, Minnesota, January 1981.

[HEVN81] Hevner, A., "Transaction Optimization Techniques in a Distributed Database System," Tech. Rep. HR-81-259 Honeywell CCSC, Bloomington, Minnesota, July 1981.

[HONE72] Honeywell Information Systems, Integrated Data Store (IDS/II) References Manual, Wellseley, Massachusetts, 1972.

[LISK79] Liskov, B., "Primitives for Distributed Computing," Proc. of The 7th Symposium on Operating Systems Principles, ACM, December 1979.

[ROSE78] Rosenkrantz, D., Stearns, R.. and Lewis, R., "System Level Concurrency Control for Distributed Database Systems," ACM Transactions on Database Systems, Vol. 3, No. 2, June 1978.

[SKEE82] Skeen, D., "A Quorum-Based Commit Protocol," Proc. of the 6th Berkeley Workshop on Distributed Data Management and Computer Network, February 1982.

[WEEL80] Weeldreyer, J., "Structrual Aspects of the Entity-Category-Relationship Model of Data," Tech. Rep. HR-80-251, Honeywell CCSC, Bloomington, Minnesota, March 1980.

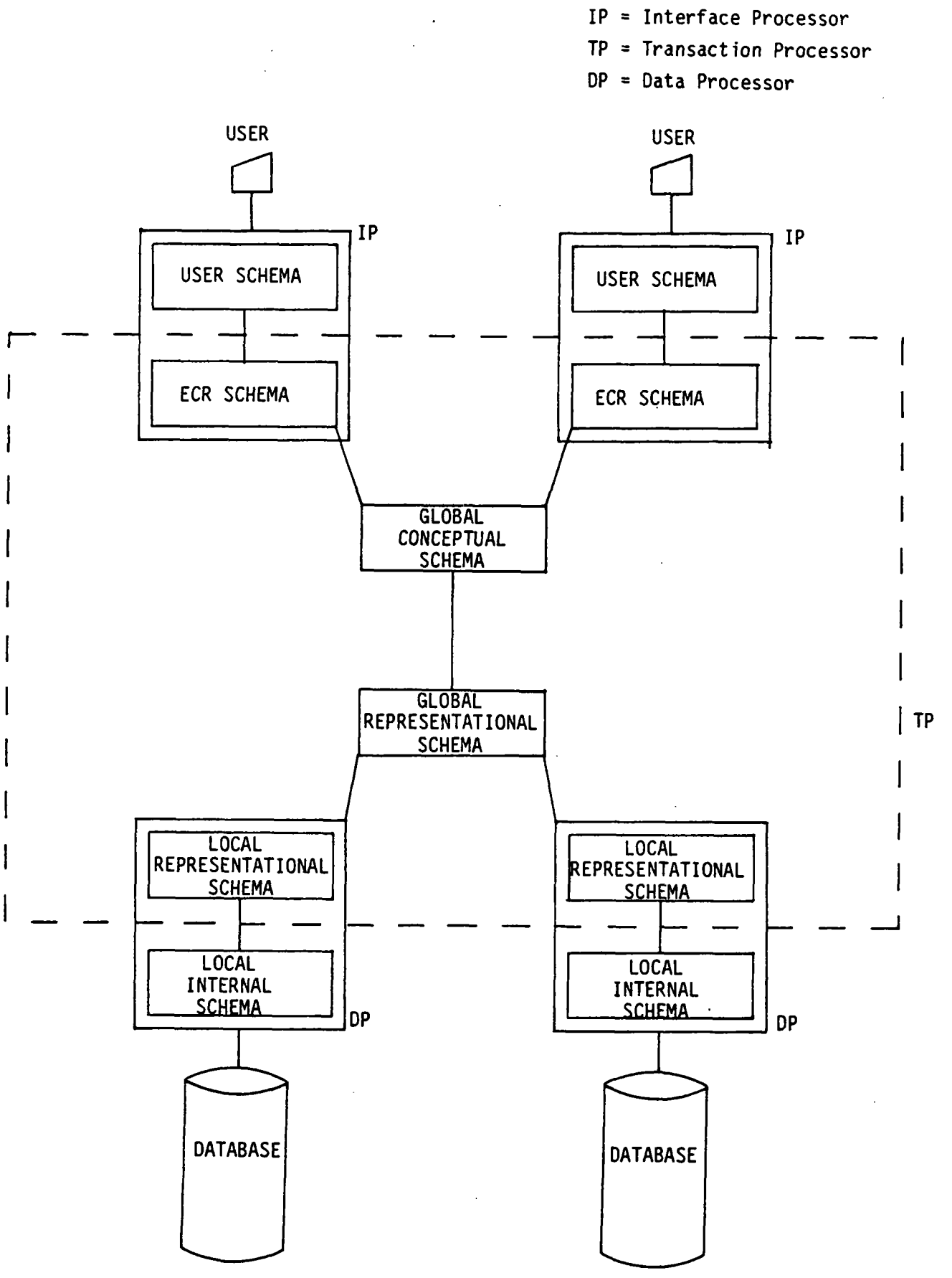


Figure 1: 6-Schema Architecture

SCOOP : a System for COOPeration between existing heterogeneous distributed data bases and programs

S. Spaccapietra[1], B. Demo[2], A. DiLeva[2], C. Parent[1]

[1]Institut de Programmation
Université Pierre et Marie Curie (Paris 6)
4, Place Jussieu
75230 Paris Cedex 05
France

[2]Istituto di Scienze dell'Informazione
Università di Torino
C. M. d'Azeglio, 42
10125 Torino
Italia

INTRODUCTION

The goal of the SCOOP project is to investigate the mapping algorithms which are needed to build a cooperation-DDBMS, that is a distributed system which allows for full integration of existing data bases and application programs. SCOOP's research fits therefore into a rarely investigated field of DDBMS design. It is hoped that it will allow a large set of applications (disregarded by actual DDBMS) to benefit by the current advance in the area of distributed data bases.

A cooperation-DDBMS should be able to:

- keep the existing data by integrating them into the DDB with (ideally) no change at all;
- keep the existing application programs and let them run as before while addressing the local DB through the DDBMS;
- use the local DBMSs for local data management (to reduce software development cost);
- offer the other facilities currently provided by the existing DBMSs.

Figure 1 illustrates the idea of a heterogeneous cooperation-DDBMS. To build such a system, new methods and tools, specifically taking into account the distributed environment, have to be designed to perform the needed translation of data structures and programs from the external level to the conceptual level within the DDBMS. It is the purpose of the project SCOOP (System for COOPeration), currently on progress at the University of Paris 6 in cooperation with the Turin University, to investigate such new methods and tools.

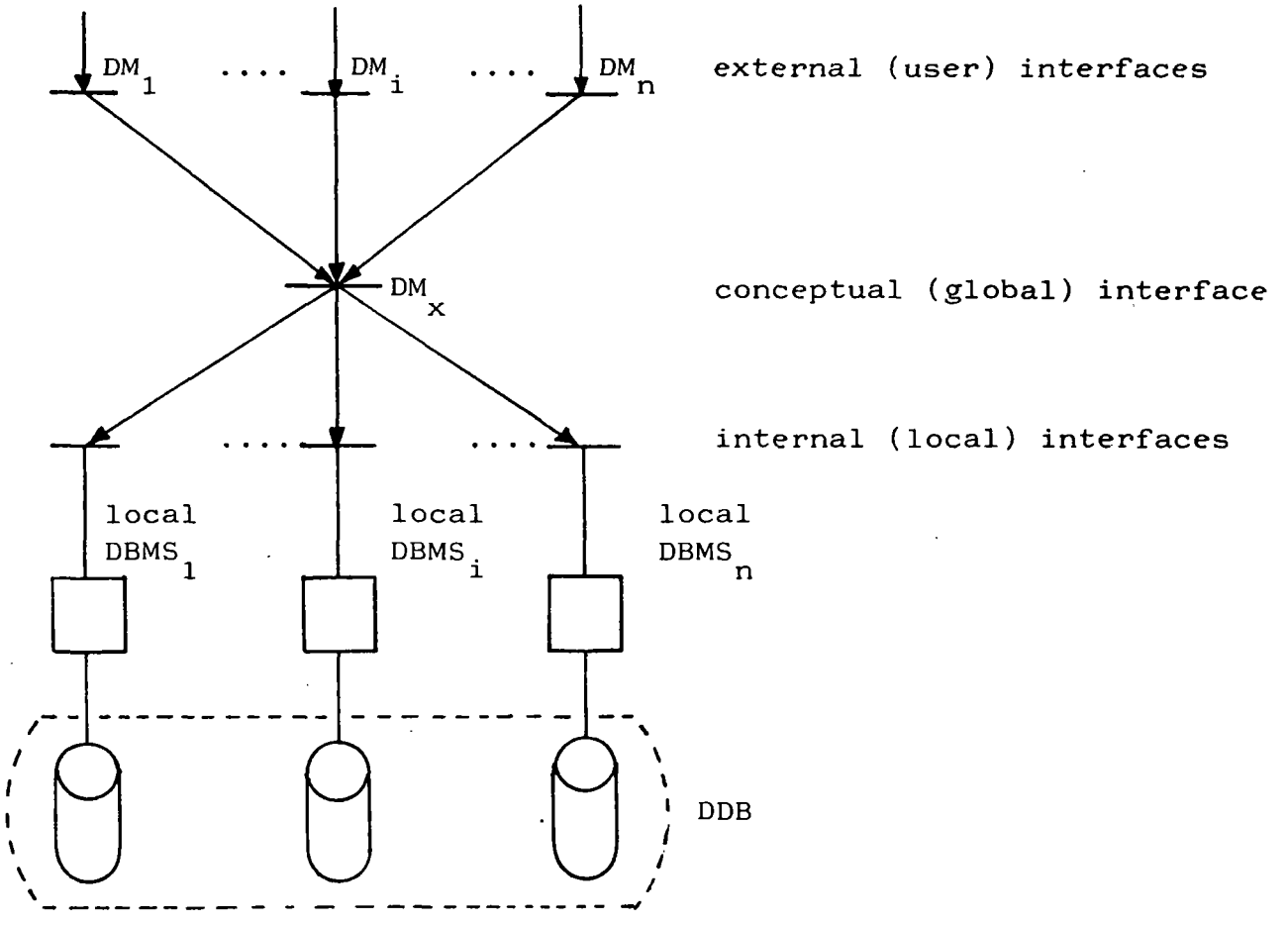


Figure 1 : Skeleton of a heterogeneous cooperation-DDBMS
(DM = data model)

SCOOP ARCHITECTURE

The overall project architecture is illustrated in figure 2; for a more detailed discussion of the architecture the reader may refer to (Spaccapietra 81).

The SCOOP project concentrates on the mappings between the external and the conceptual level in the DDBMS. For accessing data through the network, SCOOP relies on the POLYPHEME prototype, which is responsible for the overall management of the DDB (Adiba 80). The interface between SCOOP and POLYPHEME is defined as a relational DDL, allowing the description of 3NF schemata, and an associated QUEL-like DML. Consequently, an additional mapping, from the conceptual level to this interface, has been included in SCOOP.

Input to SCOOP are the user application programs, which may be written in any of the DMLs supported by existing DBMSs; the data referred to by a program are described by a global external schema (ges_i) expressing the user's view of the DDB.

The SCOOP system is made up of six different types of module: the first three are devoted to schema translation, while the next three perform program translation. These modules are:

- schema translation modules (ST_i): these translate an user schema (ges_i) into an equivalent, external schema (ees_i) expressing the same semantics through the conceptual data model. This corresponds to a first step (model mapping) in the mapping process from the external to the conceptual level. There is one such module for each DDL supported by SCOOP.

- schema integration module (SI): this performs the second step (semantic mapping) in the external to conceptual mapping. It builds the semantic mapping linking the ees_i to the (global) conceptual schema gcs describing the DDB and integrating all of the ees_i. As the ees_i and the gcs use the same data model, only semantic trasformations are perfomed by SI.

- schema conversion module (SC): this module performs a model mapping to convert the conceptual schema into the equivalent conceptual schema (ecs) used as interface to POLYPHEME. Ecs is a 3NF relational schema.

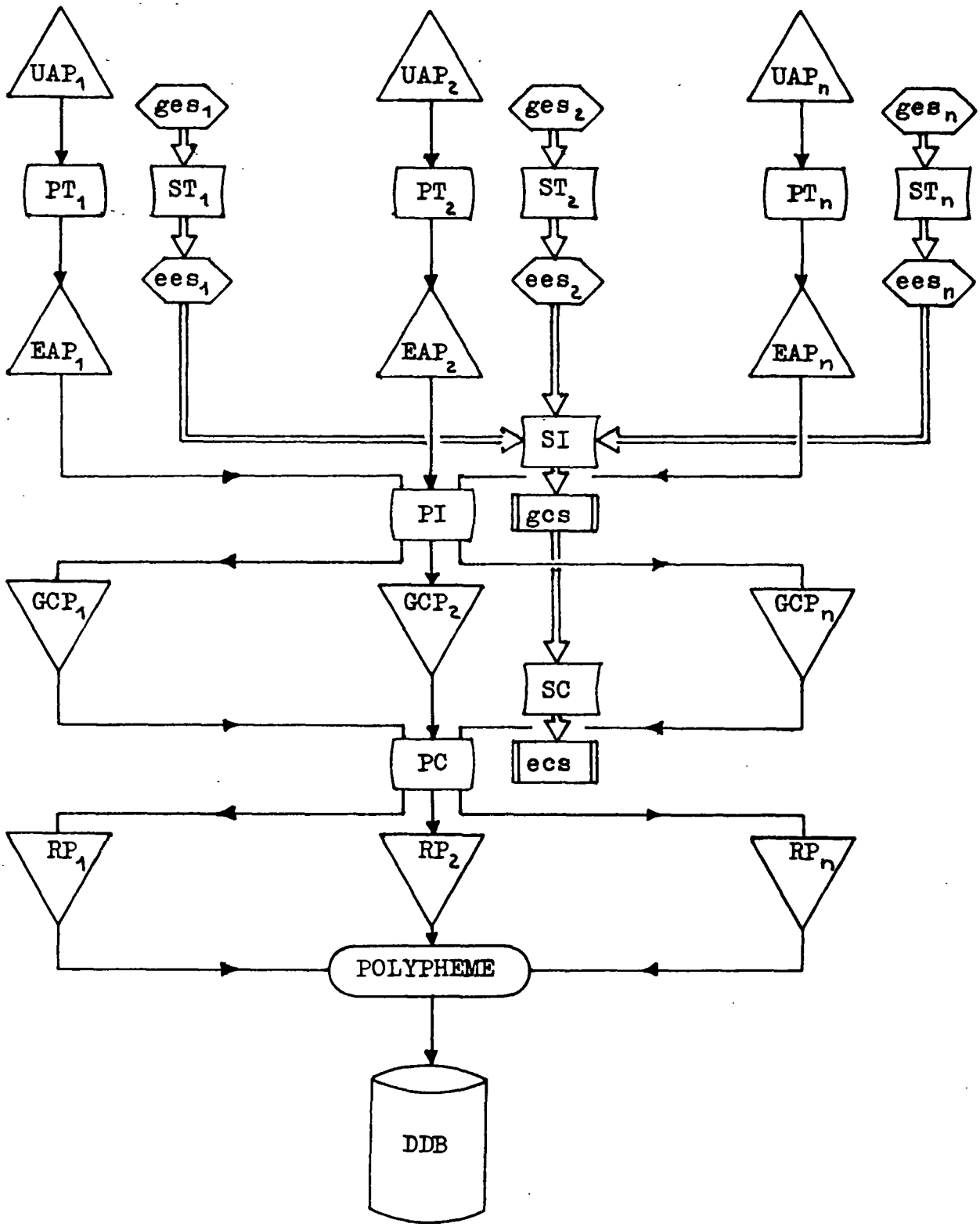


Figure 2 : SCOOP architecture

- program translation modules (PT_i): these modules translate user application programs (UAP_i) into equivalent programs (EAP_i) in which data manipulations are stated in the DML defined for the conceptual level and refer to the associated ees_i. There is a PT_i for each DML/DDL couple supported by SCOOP.

- program integration module (PI): this module translates the equivalent application programs (EAP_i) into global conceptual programs (GCP_i) by transforming references to the ees_i to references to the gcs.

- program conversion module (PC): this last module translates the GCP_i into their equivalent relational programs (RP) which will interact with POLYPHEME.

The above architecture is based on two assumptions: the first one is that the conceptual data model is different from the relational model supported by POLYPHEME, which introduces the need for the SC and PC modules. The second one is that the external to conceptual mapping is split into its two separate components: model mapping and semantic mapping.

The definition of the conceptual interface certainly is the most sensitive point in the design of a system like SCOOP. A complete definition includes the choice of a data model as well as the choice of a DDL and a DML. For the data model, the Entity Relationship proposal has been selected, mainly because it is reasonably good at expressing the semantics of the real world and easy to understand and use. A discussion on the criteria of this choice is in (Spaccapietra 82).

For the DML, the specific criteria due to the distributed environment suggest the development of a new DML. A complete description of the SCOOP-ER DML working definition will be available in an incoming paper (Parent 82).

SCOOP's architecture relies on the definition of a set of mappings: the CODASYL versus ER schema mapping process is described in (Perez de Celis 81). Some work has been done on the schema and program conversion techniques for mapping the conceptual ER interface into POLYPHEME's relational interface (Belfar 81).

Current research is mainly on the CODASYL translation and integration mapping and should result in the implementation of a first software prototype of SCOOP translation methods in 1983 (Demo 81).

REFERENCES

Adiba 80 Adiba M. et al: "POLYPHEME: An experience in distributed database system design and implementation", in Distributed Data Bases, North-Holland, 1980.

Belfar 81 Belfar K. : "Conversion de données et de programmes d'un modèle Entité-Relation à un modèle relationnel", TR SCOOP 81.4, Institut de Programmation, 1981.

Demo 81 Demo B. Spaccapietra S. : "A CODASYL COBOL statements classification for conversion into an assertional DML", TR.SCOOP 81.6, Institut de Programmation 1981.

Parent 82 Parent C., Demo B., Spaccapietra S. : An Entity-Relationship DML for distributed DBMSs, TR.SCOOP 82.2, Institut de Programmation 1982 (in progress)

Perez de Celis 81 Perez de Celis C. : Traduction du LDD CODASYL à un LDD Entité-Relation, TR.SCOOP 81.3, Institut de Programmation, 1981.

Spaccapietra 81 Spaccapietra S. et al : "An approach to effective heterogeneous databases cooperation", in Distributed Data Sharing Systems, North-Holland, 1982.

Spaccapietra 82 Spaccapietra S. et al. : "SCOOP: a system for integrating existing distributed heterogeneous data bases and application programs", TR.SCOOP 82.1, Institut de Programmation, 1982.

PERFORMANCE ANALYSIS OF DISTRIBUTED DATA BASE SYSTEMS*

by

Michael Stonebraker, John Woodfill, Jeff Ranstrom, Marguerite Murphy
Joseph Kalash, Michael Carey and Kenneth Arnold

DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
UNIVERSITY OF CALIFORNIA
BERKELEY, CA.

ABSTRACT

In this paper we discuss the design of Distributed INGRES and the performance testing that is planned for it. We also give initial benchmark data for the system. In addition, we discuss analytic and simulation studies which are in progress and implementation difficulties we have faced.

1. INTRODUCTION

There have been a considerable number of algorithms developed to support distributed relational data bases in the areas of concurrency control, crash recovery, support of multiple copies of data, and command processing.

At present there is little concrete knowledge concerning the performance of such algorithms. Previous work has been based exclusively on simulation, *e.g.* [RIES79, GARC79, LIN81] or formal modeling, *e.g.* [GELE78, BERN79]. It is the basic objective of the Distributed INGRES project to provide empirical results concerning the performance of alternate algorithms.

In Section 2 we discuss the current state of Distributed INGRES. Then, in Section 3 we present initial benchmark data on the running system. Section 4 discusses the additional benchmarks that are planned while Section 5 and 6 describe some simulation and analytic studies that are underway. Lastly, Section 7 comments on the implementation difficulties that we have faced.

2. DISTRIBUTED INGRES

Distributed INGRES operates in a hardware environment consisting of a collection of DEC VAX 11/780s and 11/750s all running the UNIX operating system. In fact, all run 4.2BSD, a version of UNIX enhanced at Berkeley with paging and numerous program development tools. As of September 1982 there are 5 11/780s and 5 11/750s connected by a 3Mbit ETHERNET purchased from Xerox. The 4.2BSD software has been extended to support remote interprocess communication and remote execution of a process. Hence, one can spawn a process on a remote machine and then do interprocess communication with that process as if it were on the same machine.

---

* Research Sponsored by the Air Force Office of Scientic Research under Grant number (22544), by the Navy Electronics Systems Command under Contract number (25990) and by the Army Research Office under Grant number Z.

Distributed INGRES has been described in [EPST78] and is operational with many of its features at this time. It consists of a master INGRES process which runs at the site where the command originated and slave INGRES processes at each site which have data involved in the command. The master process does parsing, view resolution and creates an action plan to solve the command using the fragment and replicate technique. The slave process is essentially one-machine INGRES [STON76] with minor extensions and the parser removed. The coordinator and slaves communicate over the 4.2BSD interprocess message system.

Distributed INGRES supports fragments of relations at different sites. For example, one can distribute a relation EMP as follows:

```
create EMP (name = c10, salary = i4, manager = c10,
            age = i4, dept = c10)

range of E is EMP
distribute E at Berkeley WHERE E.dept = "shoe"
            at Paris   WHERE E.dept = "toy"
            at Boston  WHERE E.dept != "toy" and
                             E.dept != "shoe"
```

Berkeley, Paris and Boston are logical names of machines which are mapped to site addresses by a table lookup. The distribution criteria is assumed to partition the EMP relation and is not currently checked for this property by Distributed INGRES software. A one site relation is a special case of the above distribute command, *e.g.*

```
distribute ONE-SITE at Berkeley
```

At the current time all QUEL commands are processed correctly for distributed data with the exception of aggregates. For example, it is acceptable to perform the following update:

```
range of E is EMP
replace E(dept = "toy") where e.salary > 10000
```

This command will be processed at all three sites where fragments of the EMP relation exist. Moreover, all qualifying tuples must have an update performed and their site location may have to be changed.

A two phase commit protocol is implemented [GRAY78]. Hence, a "ready" message is sent from the slaves to the master when they are prepared to commit the update. If there are tuples which change sites, they are included with the ready message. The master can then process the tuples from all sites and redistribute them. This redistribution is accomplished by piggybacking the tuples onto the commit message when it is sent out. Optionally, a three phase commit protocol can be used [SKEE82] for added reliability. In this case the above redistribution is handled in phase two.

When a command spans data at multiple sites, a rudimentary version of the "fragment and replicate" query processing strategy is implemented. We illustrate this module by example. Suppose a second relation

```
DEPT (dname, floor, budget)
```

exists at two sites as follows:

```
distribute D  at Berkeley  where D.budget > 5
          at Paris     where D.budget <= 5
```

and suppose a user submits the following query at Boston:

range of E is EMP
range of D is DEPT
retrieve (E.name) where E.dept = D.dname and D.floor = 1

First, the one variable clause "D.floor = 1" is detached from the query and run at Berkeley and Paris, *i.e.*

range of D is DEPT
retrieve into TEMP (D.dname) where D.floor = 1

The original query now becomes

range of E is EMP
range of D is TEMP
retrieve (E.name) where E.dept = D.dname

Data movement must now take place to satisfy the query. One relation (say TEMP) is chosen to be replicated at each processing site. Hence, both Berkeley and Paris send their portion of the TEMP relation to each site which has a fragment of EMP. The needed transmissions are:

TEMP(Paris) -> Boston
TEMP(Paris) -> Berkeley
TEMP(Berkeley) -> Paris
TEMP(Berkeley) -> Boston

At this time all three sites have a complete copy of TEMP and a fragment of the EMP relation. The above query is performed at each site, yielding a portion of the answer. As a last step each site returns tuples to the master site which displays them to the user.

Since our ETHERNET has the hardware capability to support broadcast, it is possible to perform the above four transfers by broadcasting each fragment of TEMP to the other two sites. However, the 4.2BSD operating system software does not support multicast or broadcast transmissions. Consequently, the above transmissions must take place individually and our strategy of replication may perform poorly [EPST78]. The network on which we planned to run [ROWE79] supported broadcast, and the code has not been changed.

At the moment, the relation to be replicated is chosen arbitrarily, so TEMP and EMP are equally likely to be selected for movement. A more elegant strategy is being planned.

3. INITIAL EXPERIMENTAL OBSERVATIONS

In these experiments we use a data base of employees with fields as discussed in Section 2. Our data base consists of 30,000 employee tuples, each 38 bytes in width. Our benchmark consisted of 1000 random updates of the form:

replace E (salary = K) WHERE E.name = L

For this benchmark we compare the performance of four different INGRES configurations:
a) Normal INGRES on a single site data base with a VAX 11/780 CPU
b) Distributed INGRES run on a data base that happens to reside at the site from which the benchmark originates. This site has a VAX 11/780 CPU.
c) Distributed INGRES run on a data base spread evenly over three machines, one VAX 11/780 and two VAX 11/750s. In this way exactly 1/3 of the updates are performed at each of three sites. Moreover, the benchmark was submitted in three job streams one at each site so as to avoid forcing a single site to be "master" for every command. Consequently, the master running at each site will discover an update which is equally likely to be processed at any of the three sites.

In this case we report statistics for each site individually as well as a summation.
d) A computation called 3*780. This row is obtained by multiplying the 11/780 numbers from c) by three. Since one-third of the total work is performed at each site, this is an estimate of the resources which would be consumed if the benchmark had been run on three VAX 11/780s.

Table 1 gives three measures for each system, elapsed time, CPU time spent in application code, and CPU time spent inside the operating system.

The conclusion to be drawn from Table 1 is that Distributed INGRES is about 20 percent slower than normal INGRES when run on a local data base. This time is largely the extra overhead which Distributed INGRES must spend examining the distribution criteria and ascertaining that each of the commands is a local one. This checking is performed at run time in the current implementation; however, a smarter implementation would perform most of it at compilation time. The second source of overhead cannot be diverted to compile time. Each tuple which is updated must be checked against the distribution criteria to ensure that it is not being updated in a manner that would physically change its location.

Second, note that 3*780 Distributed INGRES uses 20 percent more CPU time than Distributed INGRES run on a local data base and 47 percent more CPU time that Normal INGRES. This appears to be the overhead of communication with a non-local data base. However, it could not possibly run slower than the 13:34 time reported for three site Distributed INGRES. Hence, it cuts elapsed time by at least 50 percent compared to Distributed INGRES on a one-site data base and 40 percent compared to Normal INGRES.

Benchmark 1 results in 522,880 bytes being transferred across the network in case c) and uses less than two percent of the available bandwidth. It appears that a large number of machines could be added to ETHERNET before there were any bandwidth limitations. Also, as long as the workload partitioned evenly, total CPU time should remain a constant and be divided among a larger and larger collection of machines, resulting in a throughput essentially linear in the number of machines.

	user time	system time	elapsed time
Normal INGRES	7:34	3:04	22:34
Distributed INGRES - local data base	9:06	3:53	26:57
Distributed INGRES - three sites			
11/780	3:43	1:30	12:43
11/750	5:28	2:16	13:34
11/750	5:48	2:13	13:22
total	14:59	5:59	-
3*780	11:09	4:30	-

Table 1

4. FURTHER EXPERIMENTATION

We propose to run a variety of benchmarks in our environment. We propose to vary the number of sites from which transactions originate, how many sites have the data required for individual commands and how much data is required to be moved between sites. The basic objectives are the following:

a) Network limitations

We speculate that it will be impossible to saturate our 3Mbit network. The reason is that CPU overhead to manage the network and do local data base processing is likely to saturate all computers on a reasonable size network before this bandwidth is achieved. We propose to measure the maximum bandwidth which our benchmark consumes. The result of this test will give insight into whether network delay or bandwidth is ever a significant issue in our environment. Moreover, we propose to explore under what circumstances a distributed DBMS can use more than 50Kbits of bandwidth. This will test whether our software could saturate a long haul network such as the ARPANET. This test will shed light on whether semi-join tactics which minimize data transmissions are desirable in distributed environments.

b) Message Limitations

We speculate that the operating system cost of sending and receiving messages may be a significant factor in distributed data base performance, and propose to test this hypothesis by direct measurement. If so, a distributed DBMS should attempt to package large messages.

On the other hand, if the operating system cost for messages is not significant, then we will have discovered that the entire network subsystem is not a bottleneck in a distributed DBMS. This has great impact on the criteria to be optimized by the query processing algorithm.

c) CPU Saturation

We expect that many benchmarks will saturate all CPUs which are involved in command processing and this will be the fundamental limitation in a distributed DBMS. If so, a query processing algorithm should schedule the work over as many machines as possible.

d) Uneven Work Distribution

Our simulation of a similar environment [MCC081] showed that an uneven workload distribution among the machines caused substantial performance degradation. We noted that statistical fluctuations in a uniformly distributed workload could easily cause the command processing loads at the various sites to become unbalanced. In this case response time for a distributed transaction became the response time of the processing site with the heaviest load. This site was slowest to respond and the transaction could not be completed until this site finished.

Also, we found that an uneven work distribution, once created, tended to persist for a long period of time. Hence, poor response time also tended to persist. A similar phenomenon has been observed in the locking subsystem of System R [BLAS79].

We plan to measure to what extent this uneven workload phenomenon surfaces in a benchmark of uniformly distributed work. If it is sizeable, then a query processing algorithm should make rebalancing the workload its optimization criteria.

5. CONCURRENCY CONTROL

The experiments sketched above should shed light on query processing and crash recovery algorithms. In addition, we expect to experiment with a variety of concurrency control schemes. Unfortunately, there are twenty or more schemes which have been proposed. Instead of attempting to implement all twenty in Distributed INGRES (which was never designed with schemes other than locking in mind), we are proceeding by a combination of theoretical analysis and simulation.

We have proposed an abstract model of concurrency control algorithms within which we can address the performance tradeoffs of various popular schemes. The model facilitates comparisons of the CPU overhead, storage overhead, concurrency characteristics, and message overhead of alternative schemes. A report on this analysis is nearing completion [CARE82].

In order to validate the conclusions of the model and to offer further insight we have also written a simulator of distributed concurrency control schemes. Experimentation with this simulator will commence shortly. We intend to validate the simulator by comparing its results for the Distributed INGRES locking scheme with actual experimental data.

6. DISTRIBUTED ARCHITECTURES

An important aspect of any distributed data base system is sizing considerations. I/O subsystems, CPUs and networks must be balanced to achieve maximum throughtput. Moreover, the topology of the network may be a consideration. We are constructing a second simulation model which can evaluate alternate distributed architectures. Using this model we hope to experiment with environments which are not easily tested in our VAX/ETHERNET environment.

7. IMPLEMENTATION PROBLEMS

In this section we mention a few of the difficulties that we have faced in the implementation.

1) Distributed Debugging

Attempting to remove the bugs from distributed programs has proved to be a frustrating and slow process. Programs which run on one machine pretending to be several do not usually run on several machines. Debugging tools for distributed environments are very primitive.

2) Machine Time

Attempting to obtain stand-alone time on a substantial collection of machines in order to perform experiments has been difficult. The social problem of obtaining cooperation from multiple independent system administrators has proved taxing.

3) Limitations on Operating System Parameters

Distributed INGRES requires a large number of open files and connections to numerous cooperating processes. Most machines on which we try to run are not configured with sufficient maximum numbers of these objects. Moreover, most system administrators refuse to reconfigure their systems to rectify the situation. As a result we must treat file descriptors and connections as a scarce resource and allocate them to tasks carefully.

4) Code Complexity

Distributed INGRES is about 1.7 times a large as normal INGRES. It has been substantially harder to design and code than any of us realized at the outset.

5) Connection Topology

A distributed data base system has a "master" and "slaves" as noted above. However, when data movement is required, a "receptor" must be activated at the receiving site. Additionally, when tuples change sites, they must be sent from a slave to the master who sorts them and redistributes them. The slave must be prepared to accept both commands and data from a master. Lastly, a user can interrupt the master which must reset all slaves and kill all receptors. Ensuring that each process is "listening" to the correct connection under all circumstances has been difficult. We have had considerable difficulty managing a complex connection topology.

6) Boredom

Distributed INGRES has been in development since 1979. Much of that time has been spent in "wait state" awaiting operating system support for networking. We have always been in the position of either "waiting a few months for the promised arrival of general facilities" or "spending a few months on an ad-hoc implementation of special facilities which would hopefully be thrown away". We have always chosen the former; and consequently, wait state has been frustrating. Moreover, a project which shows little noticeable progress results in boredom for the implementation team.

REFERENCES

[BERN79] Bernstein, P. and Chiu, D., "Using Semi-joins to Solve Relational Queries", Computer Corp. of America, Cambridge, Mass., Jan. 1979.

[BLAS79] Blasgen, M. et. al., "The Convoy Phenomena", Operating Systems Review, April, 1979.

[CARE82] Carey, M., "An Abstract Model of Data Base Concurrency Control Algorithms" (in preparation).

[EPST78] Epstein, R., et. al., "Distributed Query Processing in a Relational Data Base System," Proc. 1978 ACM-SIGMOD Conference on Management of Data, Austin, Texas, May, 1978.

[CARE82] Carey, M., "A Formal Model of Concurrency Control Systems" (in preparation).

[GARC79] Garcia-Molina, H., "Performance of Update Algorithms for Replicated Data in a Distributed Data Base," PhD Thesis, Stanford University, Computer Science Dept, June 1979.

[GELE78] Gelenbe, E. and Sevcik, K., "Analysis of Update Synchronization for Multiple Copy Data Bases," Proc. 3rd Berkeley Workshop on Distributed Data Bases and Computer Networks, San Francisco, Ca., February 1978.

[GRAY78] Gray, J., "Notes on Data Base Operating Systems," in Operating Systems: An Advanced Course, Springer-Verlag, 1978, pp393-481.

[LIN81] Lin, W., "Performance Evaluation of Two Concurrency Control Mechanisms in a Distributed Data Base System," Proc. 1981 ACM-SIGMOD Conference on Management of Data, Ann Arbor, Mich., May 1981.

[MCCO81] McCord, R., "Sizing and Data Distribution for a Distributed Data Base Machine," Proc. 1981 ACM-SIGMOD Conference on Management of Data, Ann Arbor, Mich., April 1981.

[RIES79] Ries, D., "The Effects of Concurrency Control on Data Base Management System Performance," Electronics Research Laboratory, Univ. of California, Memo ERL M79/20, April 1979.

[ROWE79] Rowe, L. and Birman, K., "Network Support for a Distributed Data Base System", Proceedings of the Fourth Berkeley Workshop on Distributed Data Management and Computer Networks, August, 1979, San Francisco, California.

[SELI80] Selinger, P. and Adiba, M., "Access Path Selection for a Distributed Relational DBMS," Proc. International Conference on Data Base management, Aberdeen, Scotland, July 1980.

[SKEE82] Skeen, D., "A Quorum-Based Commit Protocol," Proc. 6th Berkeley Workshop on Distributed Data Bases and Computer Networks, Pacific Grove, Ca., Feb 1982.

[STON76] Stonebraker, M. et. al., "The Design and Implementation of INGRES," TODS 2, 3, September 1976.

DATABASE WEEK

SIGMOD83 Plus Special Sessions on Databases for Business Applications and Engineering Design

May 23-26, 1983, Hyatt Hotel, San Jose, California

Sponsored by: ACM SIGMOD, ACM SIGBDP, IEEE TC on VLSI, IEEE TC on Design Automation, IEEE TC on Database Engineering

In cooperation with: ACM SIGIR, Institut National De Recherche en Informatique et en Automatique (INRIA)

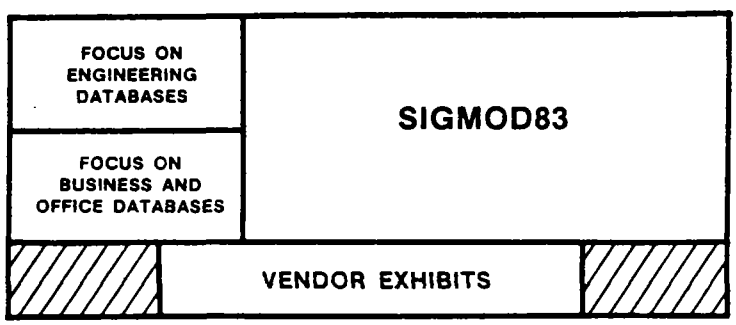
TOPICS OF INTEREST

ENGINEERING DATABASES: Relational database support for CAD/CAM, database for VLSI design, graphics data management, database machines for engineering data, extensions of database systems for supporting design.

BUSINESS AND OFFICE DATABASES: Non-formatted data (speech, text, image), databases for graphics, office data requirements, databases for workstations and personal computers, databases for decision support, experience with database systems, dictionary design, database system selection and administration, database support for expert systems, centralized versus distributed databases.

SIGMOD83: Database techniques and theory applicable to all areas, distributed database systems, database models, systems issues (concurrency, locking, transaction management), database machines, query systems, query optimization and compilation, database languages, data sharing for multiprocessors, and related topics.

— MAY 23-26 —



General Chairperson
Robin Williams
IBM Research, K55/281
5600 Cottle Road
San Jose, CA 95193

SIGMOD U.S. Chairperson
David J. DeWitt
Computer Sciences Dept.
University of Wisconsin
1210 W. Dayton Street
Madison, WI 53706

SIGMOD European Chairperson
Georges Gardarin
INRIA
B.P. 105
Domaine de Voluceau-Rocquencourt
78150 Le Chesnay, France

Engineering Databases Chairperson
Raymond Lorie
IBM Research, K54/282
5600 Cottle Road
San Jose, CA 95193

Business and Office

Database Chairperson
Eric Carlson
Convergent Technologies
3055 Patrick Henry Dr.
Santa Clara, Calif. 95050

Exhibits Chairperson
Paula Hawthorn
Lawrence Berkeley Laboratory
CSAM 50B/3215
Berkeley, CA 94720

Call for Papers and Participation

Second International Workshop on Statistical Database Management

27-29 September 1983, Los Altos, California

Sponsors
Lawrence Berkeley Laboratory
University of California

U.S. Department of Energy

In Cooperation With
Association for Computing Machinery, SIGMOD

American Statistical Assoc.
Statistical Computing Section

General Chairperson
John L. McCarthy
Lawrence Berkeley Laboratory

Program Chairperson
Roy Hammond
Statistics Canada

Program Committee
Rick Becker
Bell Telephone Laboratories
Francis Chin
University of Alberta
Ivor Francis
Cornell University
Jerome Friedman
Stanford Linear Accelerator Center
James Gentle
International Mathematical and Statistical Libraries, Inc.
Ron Helms
University of North Caroline
David Hoaglin
Harvard University
Gregory A. Marks
Inter-University Consortium for Political and Social Research
Wes Nicholson
Pacific Northwest Laboratory
Gordon Sande
Statistics Canada
Diane C. P. Smith
Computer Corp. of America
Peter Stevens
Bureau of Labor Statistics
Stanley Su
University of Florida
Harry Wong
Lawrence Berkeley Laboratory

Conference Coordinator
Peggy Little
Lawrence Berkeley Laboratory
Building 80C
Berkeley, CA 94720
(415) 486-6386
FTS: 451-6386
Arpanet: john@lbl-unix

Workshop Program

This limited attendance workshop will bring together computer scientists, statisticians, system designers, and others to discuss current work on statistical database management. Working groups of five to fifteen participants will meet daily to discuss and draft position papers on individual topics. Plenary sessions will include special presentations, discussion of selected topics, and working group reports. Individual research reports, issue outlines, papers, and biographical sketches will be circulated prior to the workshop.

Topics

Like its highly successful predecessor in December 1981, this workshop will address research and implementation issues including, but not limited to, the following areas of statistical database management:

Meta-data	**Storage and Access**	**Applications**
conceptual models	data structures	scientific experiments
schema definition	compression methods	medical records
data dictionaries	security and privacy	economic time series
self-describing files	distributed databases	data analysis
User Interfaces	**Comparative Analysis**	**Hardware**
languages	performance evaluation	database machines
software tools	interface experiments	storage technology
interactive requirements	benchmark standards	microprocessors

Participation by Invitation

The Program Committee will invite 50 to 100 people to the workshop, based on written research reports, issue outlines, and papers. Contributions should be 1000 to 5000 words in English (preferably in IEEE format), with a separate page containing an abstract and a biographical sketch of each author (no more than 100 words each). Send five (5) final copies by 1 March 1983, to

Roy Hammond, Program Chairperson
Statistics Canada, EPSD
R. H. Coats Building -- 13th Floor
Tunney's Pasture, Ottawa Canada K1A 0T6
(telephone 613 995-3973)

The Program Committee will notify authors by 1 May 1983.

Location, Cost, and Accommodations

The workshop will be held at a 50 acre hilltop retreat center in Los Altos, California, about forty miles south of San Francisco. The cost will be approximately $250 per person, including registration plus three full days room and board. There will be a special registration-only rate for full-time students.

Important Dates

Submission Deadline	1 March 1983
Acceptance Notification	1 May 1983
Final Version Due	1 July 1983
Proceedings Mailed	1 September 1983
Workshop	27-29 September 1983

Further Information

For additional information, fill in and mail the attached response card, or contact the LBL Conference Coordinator listed at left.

Response card *(please mail to address on reverse side)*

Name ..... **Telephone:** .....

Organization .....

Address .....

City, State, Zip Code and Country .....

please check all of the following that apply:

___ I intend to submit a research report
___ I intend to submit a working group issue outline
___ I intend to submit a paper
___ I would like to help organize a working group
___ Not sure I can participate, but please keep me informed.

Subject of paper, report, or outline .....

ER APPROACH

3RD INTERNATIONAL CONFERENCE ON ENTITY-RELATIONSHIP APPROACH

CALL FOR PAPERS

October 6-8, 1983 Anaheim, California

Major Theme: The Use of Entity-Relationship Concept in Software Engineering

General Conference Chairperson
Dr Carl G Davis
Ballistic Missile Defense Advanced Technology Center
P O Box 1500
Huntsville, AL 35807

Conference Chairpersons
Mr Laszlo A Belady
IBM
Old Orchard Rd
Armonk, NY 10504

Professor Donald R Shuttleff
Department of Computer Science
University of Missouri-Columbia
Columbia, MO 65211

Program Chairpersons
Professor Raymond T Yeh
Department of Computer Science
University of Maryland
College Park, MD 20742

Professor Peter A Ng
University of Missouri-Columbia

ER Conference Steering Committee Chairperson
Professor Peter P Chen
Graduate School of Management
UCLA
Los Angeles, CA 90024

Program Committee Members

Mack W Alford	USA	Sung Y Bang	Korea
Carlo Batini	Italy	Catriel Beeri	Israel
John L Berg	USA	Paul K Blackwell	USA
Janis A Bubenko	Sweden	David T Chen	USA
T C Chiang	USA	Peter deJong	USA
Al Dale	USA	Dennis W Fife	USA
Andre Flory	France	Robert A Frayley	USA
K S Fu	USA	A L Furtado	Brasil
L S Hsu	Singapore	Rowland Johnson	USA
Stephen R Kimbleton	USA	R C T Lee	China
Alberto Mendelzon	Canada	William E McCarthy	USA
Ephraim R McLean	USA	John Mitchell	USA
Erich J Neuhold	Germany	Jurg Nievergelt	Switzerland
Ross A Overbeek	USA	C V Ramamoorthy	USA
Douglas T Ross	USA	Nicholas Roussopoulos	USA
Hirotaka Sakai	Japan	Edgar H Sibley	USA
Arne Solvberg	Norway	Terry A Straeter	USA
Daniel Teichroew	USA	Julius T Tou	USA
John H Tsao	USA	Hubert Tardieu	France
A Min Tjoa	Austria	Jeffrey D Ullman	USA
Charles Vick	USA	Herbert Weber	Germany
James A Weeldreyer	USA	Alan B Salisbury	USA

CALL FOR PAPERS

INTERNATIONAL JOURNAL ON ENTITY RELATIONSHIP APPROACH

First Issue October 1982 (Quarterly Publication)

Topics to be covered
(1) Theory of Entities and Relationships
(2) Applications in Databases, Information Systems, Accounting, Social Sciences, Organizational Designs, Simulation, etc
(3) Case Studies

Submit Papers
Send five copies of the double spaced manuscript to
Dr Rowland Johnson
L 300
Lawrence Livermore Laboratory
Livermore CA 94550 U S A

Subscription
$20 individuals
$70 institution
+$20 extra for airmail to the U S, Canada Subscribers

For Subscription and further information contact ER Institute
P O Box 617 Saugus CA 91350 U S A

Sponsors
ER Institute
UCLA Graduate School of Management
Univ of Maryland (Dept. of Computer Science)
Univ of Missouri-Columbia (Computer Science)

This conference will bring together researchers and practitioners to focus attention upon the theory of entity relationships and its applications on software engineering.

Topics of Interest
Papers of high quality are solicited on both principles and pragmatics of ER Approach in software engineering. Topics of major interest are, but are not limited to.

Theory and Graphical Representations
- Definition of Entities, Relationships, Attributes, etc
- Modification and Extension of Entity-Relationship Models and Diagrams
- Representation of Schemes of Entity Relationships
- Semantics and Optimization Issues
- Entity-Relationship Concepts in Management Science Models
- Other Conceptual Models

Systems and Languages
- Computer Languages Based on Entities and Relationships
- Database Management Systems and Distributed Databases
- Database Schema Conversions and Translations
- Database Design Tools
- Data Dictionaries
- Database Dynamics

System Analysis and Specifications
- Software Design Techniques and Tools
- Requirements Engineering (Definition, Formulation and Analysis)
- Data Abstraction
- Software Environment
- Software Metrics
- Software Productivity
- Quality Control and Assurance
- Software Maintenance Evoluation
- Software Project Management
- Human Factors
- Organization Design
- Database Organization
- Case Studies

To Submit Your Papers
(1) Five copies (in English) of the double-spaced manuscript should be submitted by February 15 1983 to the Chairperson
Professor Peter A Ng
Department of Computer Science
University of Missouri-Columbia
Columbia, MO 65211
Tel (314) 882-4540 or 3842 (Mrs Norma Kenyon)
(2) Authors will be notified of acceptance or rejection by May 15 1983
(3) Revised papers will be due by July 15 1983

Proceedings
The conference proceedings will be published by the ER Institute Authors of all accepted papers will be expected to sign a copyright release form The hard cover proceedings of the 2nd Conference may be ordered from the ER INSTITUTE P O Box 617, Saugus CA 91350 USA ($25 prepaid $30 otherwise)