HD-DBMS : AN APPROACH TO INTEGRATING HETEROGENEOUS DISTRIBUTED DATABASE SYSTEMS IN THE OFFICE ENVIRONMENT

T.S. Narayanan, P. Goyal and B.C. Desai
Department of Computer Science
Concordia University - Loyola Campus
Montréal, Canada, H4B 1R6

ABSTRACT

Advances in communication technology and inexpensive hardware are spreading the use of distributed information systems and in particular distributed databases. There has also been a lot of interest and substantial investment in the use of personal computers. This has led to the development of microcomputer based database management systems (DBMS). These DBMS's differ in their underlying data models and their data definition, manipulation and security capabilities. As part of the development of an Office Information System a need was felt to integrate the available microcomputers and their databases. This paper provides an overview of our approach to integrate these heterogeneous microcomputer based database management systems. The innovative features incorporated in the design include :

1. To support the existence of overlapping (non-disjoint) data fragments at different nodes leading to very flexible data distribution options and thus improving locality of reference and effeciency of query processing.

2. To support the existing data models and applications; our system acts as a front-end to the existing local DBMS.

3. To exploit LAN capabilities for query optimization.

1. INTRODUCTION

Computing with microcomputers yields local data autonomy, in addition to other benefits with communication links, the individual databases can be integrated using a distributed heterogeneous DBMS. This would allow access to useful information from other users' and the corporate databases. So far no attempt has been made towards integrating the existing databases on microcomputers. Such an integration would enhance an automated office. Almost all the distributed database systems designed so far are for homogeneous distributed databases (except POLYPHEME [1], and Multibase [13,10]). In addition, some of the system designs are based on existing centralized databases (System R* developed around the system R [8], and distributed INGRES developed around INGRES [9]; other distributed database systems are entirely new (SDD-1 [7]). Users interested in migrating to these homogeneous D-DBMS may have to rebuild thei' existing database. Another aspect which has often been overlooked and ignored is overlapped (non-disjoint) fragments in data distribution [11,12]. Non-disjoint fragments are, however, important where many existing databases are being integrated into a distributed system; a common situation in office automation. In section 2 we discuss the need for overlapped partitioning (replication may be considered to be 100% overlapping. We are proposing a heterogeneous distributed DBMS (HD-DBMS) that will act as a front-end to the existing DBMS's. The system is capable of handling overlapping partitions and independent of the local DBMS. The following assumptions are made in the initial systems design.

1. The existence of a communication network. Our HD-DBMS system is being designed to use a communication subsystem which may be tailored for any communication system.

2. We assume, initially, that data incompatibilities do not exist. For example, the same data may not be stored in units of inches at one database, and in units of centimeters at another database. This problem has already been solved in distributed systems by using mapping algorithms [7,1].

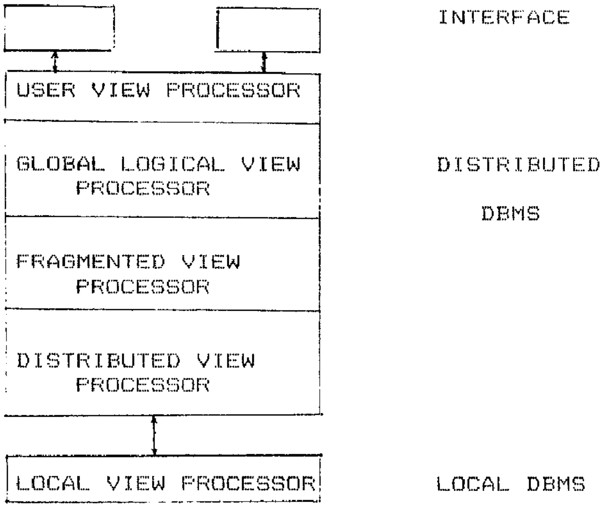

INTERFACE
USER VIEW PROCESSOR
GLOBAL LOGICAL VIEW PROCESSOR
DISTRIBUTED DBMS
FRAGMENTED VIEW PROCESSOR
DISTRIBUTED VIEW PROCESSOR
LOCAL VIEW PROCESSOR
LOCAL DBMS

Figure 1. D-DBMS Architecture.

A general multilevel architecture of the distributed database system software[2,7,9,11] is shown in Figure 1. There are five levels to this architecture, divided into two major parts -

1. Distributed database management system
2. Local database management system

EMPLOYEE

ENO	NAME	PLANT-NO	RATE
100	Bill	1	6.00
101	Jim	1	6.00
102	Mike	2	10.00
103	Houtan	2	12.00
104	Don	3	2.90
105	Steve	3	3.00

PLANT

P-NO	P-NAME
1	Ann Arbor
2	Detroit
3	New York

RAW MATERIALS

P-NO	ITEM	QUANTITY
1	P_1	500
1	P_2	100
2	P_3	940
3	P_1	75

Figure 2. Global Logical View.

Each of these levels supports a different view of the database. Tables in figure 2 and figure 3 give the global logical view and one of the users view of an hypothetical database. The user's interface is either a query processor or a host language processor.

EMPLOYEE

E-NO	NAME	PLANT-NO
104	Don	3
105	Steve	3

PLANT

P-NO	P-NAME
1	Ann Arbor
2	Detroit
3	New York

Figure 3. User View.

The fragmented and distributed view processors are characterstics of a distributed DBMS. A fragmented view defines subsets of relations known as logical fragments, e.g. one for each value of PLANT-NO (Figure 4). A fragment may contain a subset of tuples from a relation (horizontal partition), or may contain a subset of attributes from a relation (vertical partition), or a combination of both. Stored fragments are physical implementations of the logical fragments. Corresponding to one logical fragment there can be multiple physical fragments. The distributed view of the distributed DBMS gives the geographical location (node identification) of each physical fragment or complete relation. Figure 5 gives the distributed view of the example being discussed.

EMPLOYEE

E-NO	NAME	PLANT-NO	RATE	
100	Bill	1	6.00	Fragment 1
101	Jim	1	6.00	
102	Mike	2	10.00	Fragment 2
103	Houtan	2	12.00	
104	Don	3	2.90	Fragment 3
105	Steve	3	3.00	

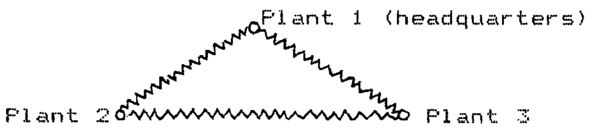
PLANT

P-NO	P-NAME	
1	Ann Arbor	
2	Detroit	Fragment a
3	New York	

RAW MATERIALS

P-NO	ITEM	QUANTITY	
1	P_1	500	Fragment A
1	P_2	100	
2	P_3	940	Fragment B
3	P_1	75	Fragment C

Figure 4. Fragmented View.



RELATION	FRAGMENT	LOCATION
EMPLOYEE	1	1
	2	1,2
	3	1,3
PLANT	a	1,2,3
RAW MATERIALS	A	1
	B	2
	C	3

Figure 5. Distributed view

For locality of reference and query optimization the local logical view should be based on the partitions of a database existing at a particular node. Figure 6 illustrates one of the local views of the example database being discussed.

EMPLOYEE

E-NO	NAME	PLANT-NO	RATE
104	Don	3	2.90
105	Steve	3	3.00

Fragment 3

PLANT

P-NO	P-NAME
1	Ann Arbor
2	Detroit
3	New York

Fragment a

RAW MATERIALS

P-NO	ITEM	QUANTITY
3	P_1	75

Fragment C

Figure 6. Local view at Plant 3.

Some of the major design issues of our proposed system are discussed in the next three sections. Design and implementation details are discussed in the subsequent sections.

2. DATA DISTRIBUTION : THE NEED FOR OVERLAPPED PARTITIONING

It is not uncommon to find overlapping data between two nodes as illustrated in Figure 7 where orders and invoices are two different relations processed at two different nodes, and having overlapping attributes. Figures 8 & 9 illustrate the necessity for overlapped data. A and B are two applications executed at two different nodes using data sets a, and b respectively. If there is an overlapping set of data (x) between these two applications, then there are two options for storing the data. The first of them is to have three non overlapping (disjoint) fragments a^1 x, and b^1 and store (a^1, x) at A, and (x, b^1) at B, with replication of the fragment x at both A and B. The second option is to store fragment a at A, and b at B with overlapping x; this is possible only if overlapping data is permitted in the design of the database.

ORDERS

Cust#	Item#	Qty
12	2	50
12	5	10
15	2	20
.	.	.
.	.	.

ITEM MASTER

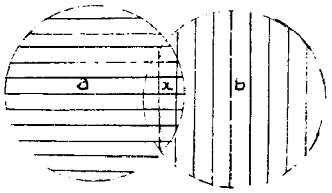
Item#	Unit Price
2	10.00
4	4.50
5	6.75
.	.
.	.

INVOICE

Customer# : 12

Item#	Qty.	Item Cost	Amount
2	50	10.00	500.00
5	10	6.75	67.50
		Total	567.50

Figure 7. Example of Overlapped Data Usage at Different Nodes.



a^1	horizontally shaded area only : a^1 a
b^1	vertically shaded area only : b^1 b
x	overlapped area only x = a b
a	horizontally shaded area $a = a^1 + x$
b	vertically shaded area $b = b^1 + x$

Figure 8.

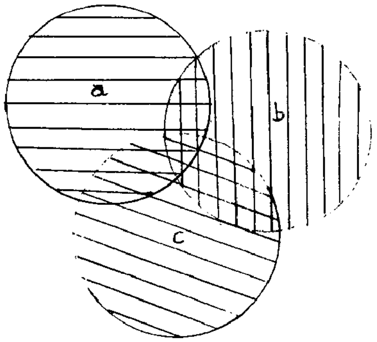


Figure 9.

Figure 9 shows the increase in the number of nonoverlapping fragments to be stored at different nodes to allow for additional applications which uses the same data. Correspondingly the number of physical fragments and their maintenance overhead increases. If overlapping is allowed, we only have to store three fragments a, b and c. In an office environment, processes at different departments may be handling such overlapping data; for example, processing a customer order involves a number of different departments. Instead of multiple fragments and replications the applications make do with overlapping data. This approach is reflected in the conventional multipart form design. Many other similar applications abound in reality.

The advantages of having overlapping data are illustrated in the following example. This example also gives some idea of how overlapping data can be handled in the global directory used in the HD-DBMS.

PART NAME	S_1	S_2	S_3	S_0	
P_1	1	0	0	1	F_1
P_2	1	0	0	1	
P_3	1	1	0	1	F_2
P_4	1	1	0	1	
P_5	0	1	0	1	F_3
P_6	0	1	0	1	
P_7	0	1	1	1	F_4
P_8	0	1	1	1	
P_9	0	0	1	1	F_5
P_A	0	0	1	1	
P_B	1	0	1	1	F_6
P_C	1	0	1	1	
P_D	1	1	1	1	F_7
P_E	1	1	1	1	

Table 1.

Example: Company x has three plants at sites S_1, S_2 and S_3 and a warehouse situated at some other site S_0, which is equidistant from the other three sites. Table 1 indicates the part's name and the plants which are using/handling that part. A '1' in the table indicates that the corresponding part is used by the column site.

If overlapping is not allowed then the data has to be partitioned into seven non overlapping segments (F_1 to F_7) as indicated in Figure 10 and are to be replicated at each site at which they are used. Query processing may require the handling of more than one fragment at a given site. Even for read operations a union of fragments has to be taken. If overlapping is allowed we only have to store a single overlapping relation at each site. The overlapping arrangement in Figure 11, gives the same or even,under certain conditions, better performance (response time, and reliability measure) as the non overlapping arrangement (Figure 10) at a reduced directory detail; the directory has only to store the details of four relations, instead of seven logical fragments and

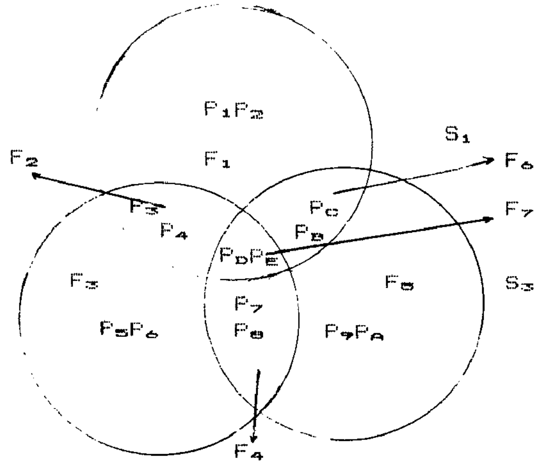


Figure 10.

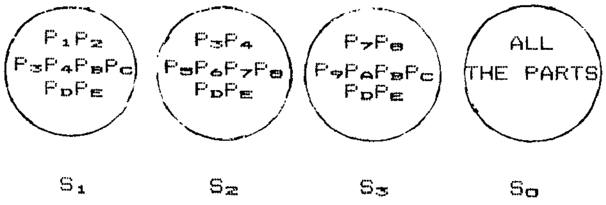


Figure 11.

TYPE	#FRAGMENTS	#REPLICATION
NO OVERLAPPING	7	19
WITH OVERLAPPING	4	4

Table 2.

nineteen physical fragments. The overlapping partitions can be handled just like horizontal partitioning for its directory details. It is obvious that the warehouse site, S_0, will have a relation containing all the parts and their details, let us call this relation at S_0 as PARTS-S0. Table 1 will be a subset of this relation and relations at the other sites can be defined using PARTS-S0 as illustrated below.

Parts relation at S_1 (PARTS-S1) =
$\sigma_{S_1=1}$PARTS-S0

Parts relation at S_2 (PARTS-S2) =
$\sigma_{S_2=1}$PARTS-S0

Parts relation at S_3 (PARTS-S3) =
$\sigma_{S_3=1}$PARTS-S0

Since there is no longer a distinction between the user's view and the logical fragments, each relation in overlapped arrangement conveys more semantics than the disjoint partitioned data.

The above gives an example of horizontal partitioning but the principles involved are the same for vertical as well as vertical-cum-horizontal partitioning.

3. DATA DIRECTORY PLACEMENT

The integration of heterogeneous databases requires that a complete global directory may not be available at all nodes. Our strategy is to broadcast a query requiring access to more details than held in the local node global directory to a suitable neighbouring node.

4. NATURE OF LOCAL DBMS

The DBMS's in use on the different microcomputers would differ in their underlying data models as well as their data definition and data manipulation capabilities. Ideally the systems would have been chosen to suit the underlying tasks but at the microcomputer database design level it is more of an individual's choice and availability. The variety in the DBMS's is unlikely to be as great as in the microcomputers on which they run. Our approach is to provide a simple integration tool for these heterogeneous DBMS's and microcomputer systems using individually tailored Translator 2 modules (Figure 12).

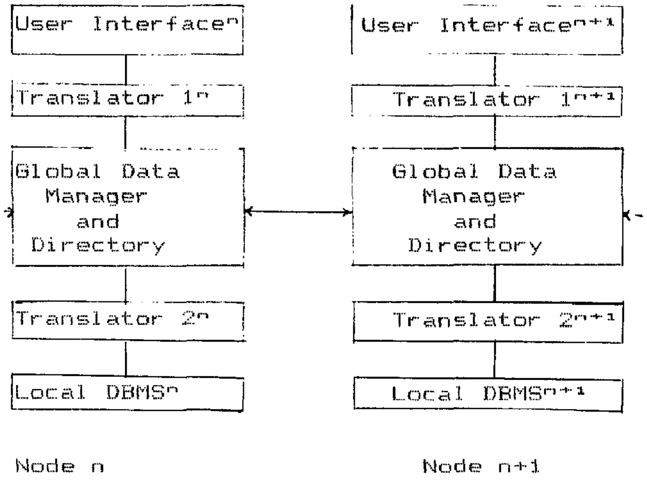


Figure 12. Overall System Architecture.

5. DESIGN OF A HETEROGENEOUS DISTRIBUTED DBMS

The proposed system architecture (Figure 13) consists of six basic components at the distributed and local DBMS levels (Figure 12).

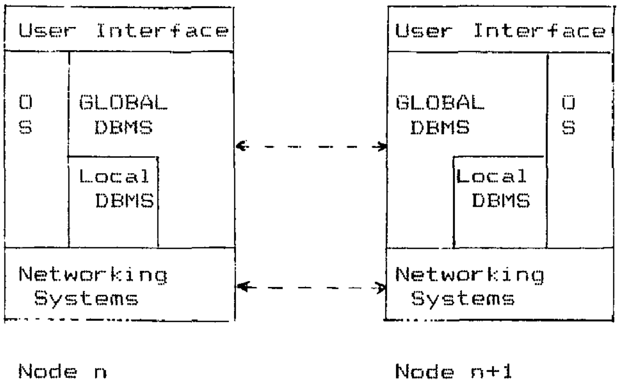


Figure 13.

The user interface provides a query processor (a superset of the local DBMS). A query entered is checked for syntactic errors at the user's interface. A syntactically correct query is passed on to the next stage - translator 1. Translator 1 translates the query to a common internal representation. The Global data manager (GDM) checks whether the query is local or if it involves data from other nodes. This checking is done by using the global directory which contains all the views and mappings. If the query is local, the internal representation of the query is once again transformed to the local query language by translator 2 and executed by the local DBMS. The data retrieved by the local DBMS is passed to the GDM; after mapping and security verification against the user access profile, the data is passed on to the user. If the query involves data from multiple nodes, the GDM coordinates with GDM's at other nodes and retrieves the data. The following paragraphs give more details about the six components.

User Interface: It is the query processor for the local user. Since the user interacts with the distributed DBMS, the query language is much more powerful than the local DBMS query language. Using this, the user can define global as well as local data. The query processor checks the syntax of the query, and if the query is correct it is passed on to the translator 1.

Translator 1: This translator has been introduced to let the user continue with the usage of his familiar but enhanced query language. The translator translates the query to a common internal representation. If there are n different micro DBMS's, we need n such translators. Since the number of micro DBMSs are limited in number, and as the number of different DBMS's in use at a given organization would be limited n is not likely to be a large number. Implementing translators is one of the bottlenecks in the design of HD-DBMS. Any number of different DBMS's can however be introduced into the system just by implementing translators and schema mapping.

Global Data Manager: Global Data Manager is the nucleus of the HD-DBMS. It performs directory management, query processing, data mapping, concurrency control and authorization checking etc. Various functional modules of the GDM are shown in Figure 14. Among the modules only the Schema mapping module needs some explanation. Schema mapping is used to suppress the differences in data models at the global view level. A number of approaches have already been considered in the literature for global schema mapping [12]. The entity relational model approach, from our initial evaluations, was found to be the best for heterogeneous database handling and its ability to cover overlapped data. The issues of overlapped data and their effects on query optimization are still under investigation.

Global Directory: Global directory is used to store details regarding all views (Figure 1), schema mapping, data mapping, protection and authorization codes and data overlap. Various data distribution strategies can be applied in directory distribution. The strategies followed in data distribution can also be applied to the distribution of the directory. Data distribution will play a major role in the directory distribution strategies. The querry processor interacts with the global directory to choose the best plan for querry evaluation.

Translator 2: Translator 2 converts the internal representation of the query into the local DBMS query. If there are n different local DBMS's we need n such translators.

Local DBMS : An existing centralized DBMS on the computer system.

User Interface	Syntax Checking
Translator 1	Translation of user querry to a common internal representation
Global Data Manager	Querry Optimization Schema Mapping, Data Definition, Data Mapping, Data Security, etc. Concurrency control and Recovery
Translator 2	Querry translation into internal local DBMS representation
Local DBMS	Actual Data Manipulation

Figure 14. Functional Specifications of System.

The following section presents three different distributed database problems and the approach to be followed in our design.

(i) Security and Protection: Once the databases are integrated they are accessible at the different nodes. If proper protection and security mechanisms are not implemented it will make the system vulnerable to unauthorized accesses; the protection and security available at the local database may not be sufficient. So another level of security is built into the system at the global directory level. Checking of the authorization is made twice at the global level; once before the access using the user's view and another after the access using the <u>field dependent protection details</u>. Sometimes data objects (tuple) are protected depending on the values taken by some attributes in the same data object (tuple) or in some other related data objects. This approach provides a uniform protection scheme at the global level irrespective of the local database management system capabilities.

(ii) Most query processing algorithms do not discuss the processing of fragments [4,14,15], others that include fragments restrict them to disjoint fragments [14]. Our query processing system is being designed to exploit overlapping fragments. As it has been repeatedly pointed out in the literature most of the fragment processing algorithms are inefficient and face problems in optimal site identification when multiple copies of data exist. Our approach would be to exploit attribute and domain details in global directories. As the system is being designed for microcomputer use effeciency is a major design goal. The overlapping arrangement of data can be handled just like fragmented data without overlapping. Because of this reason a query concerning overlapping data can be answered just like a fragmented data query. A new approach is being considered in processing queries involving replicated data. If two or more nodes have the same data the query is always answered by the best node. The query processing is generally done in two steps. First the originator node, say *k*, sends the query to a node *l* which has the global directory of interest to the query. Node *l* then divides the query into ranged subqueries based on the information about those nodes which have the relevant data. These subqueries are transmitted to these specific nodes which send their responses to the originator node *k*.

(iii) Concurrency Control: Microcomputers are usually single user systems so concurrency of transactions from the same node does not arise. However concurrent transactions from two or more nodes may try to access the same data. Such transactions are queued up and serialized. As far as the read access is concerned the DBMS can process any number of such requests concurrently. However for updates, synchronization with other nodes is necessary. This synchronization is handled by the global DBMS using global locking techniques. If a node is down the update messages are spooled and processed later. Deadlock is handled by a time-out scheme. The proposed system is also conducive to time-stamp technique. An algorithm [12,13] using simulated perfect clocks can be used.

6. IMPLEMENTATION.

The present implementation is underway on IBM PC/XT systems. Access to relational and network local database management systems is provided by the KNOWLEDGEMAN [5] and MDBS III [6] systems respectively. For the moment the interface from translator 2 to these systems is provided for by files. Later it is planned to replicate the system on to two CADMUS microcomputers running under UNIX and networked using ETHERNET. The modular design of our HD-DBMS not only allows integration of heterogeneous local DBMS's but also for different computers running under different operating systems.

A tool for automatic global schema building from underlying local database schemas is also under construction. The design includes an incremental schema integrator.

7. CONCLUSIONS.

As part of the development of an Office Information System a need was felt to integrate available microcomputers and their databases. In this paper we have presented our underlying approach. Certain problems, for example query optimization for non-disjoint fragmented and replicate data have not yet been fully solved.

10. REFERENCES:

1. Adiba, J.M., Andrade, J.M. Fernandez, F. Nguyen Toan, "An Overview of the Polypheme Distributed Database Management System", Information Processing, 1980, pp.475-479.

2. Astrahan, M.M. and Chamberlin, D.D., "System R: Relational Approach to Database Management", ACM Trans. on Database Systems, Vol.1, No.2, 1976, pp.97-137.

3. Chan, A., et.al., "Overview of an Ada Compatible Distributed Database Manager", ACM SIGMOD, 1983, pp.228-237.

4. Hevener, A., Yao, S.B., "Query Processing in Distributed Database Systems", IEEE Trans. on Software Engineering, Vol.SE-5, No.3, May 1979, pp.177-187.

5. KNOWLEDGEMAN : The Knowledge Manager, Reference Manual, Micro Data Base Systems Inc., Lafayette, Indiana 47902.

6. MDBS III Data Base Design Reference Manual, Micro Data Base Systems Inc., Lafayette, Indiana.

7. Rothnie, J.B., et.al.,"Introduction to a System for Distributed Databases (SDD-1)", ACM Transactions on Database Systems, Volume 5,No.1, March 1980, pp. 1-17.

8. "R*: An Overview of the Architecture", IBM Technical Report, RJ 3325 (40082) 12/2/81, Computer Science, 1981.

9. Stonebraker, M., Neuhold, E., "A Distributed Database Version of INGRES', Proc. Berkeley Workshop on Distributed Data Management and Computer Networks, 1977, pp.19-36.

10. Smith, J.M., et al., "Multibase-Integrating Heterogeneous Distributed Database Systems", National Computer Conference, 1981, pp.487-499.

11. Teory, T.J., Fry, J.P., "Design of Database Structures", Prentice-Hall, 1982.

12. Thomas, R.H. "A Solution to the Update Problem for Multiple Copy Data Bases Which Uses Distributed Control." BBN Report 3340 (1976).

13. Thomas, R.H. "A Majority Consensus Approach to Concurrency Control for Multiple Copy Databases". ACM Transactions on Database Systems, Vol.4,No. 2, June 1979, pp.180-209.

14. Wong, E.,"Retrieving Dispersed Data from SDD-1:A System for Distributed Databases", Proc. Berkeley Workshop on Data Management and Computer Networks, 1977.

15. Yu, C.T., Chang, C.C., "On the Design of a Query Processing Strategy in a Distributed Database Environment", ACM SIGMOD, 1983, pp.30-39.