NOTES ON DDTS

An Apparatus for Experimental Research

in

Distributed Database Management Systems

Ramez Elmasri
Cory Devor
Said Rahimi

Honeywell
Corporate Computer Sciences Center
Bloomington, Minnesota 55420
(612) 887-4356

1. Introduction

In a recent letter to CACM [DENN80], Peter Denning indicated that "experimental science classifies knowledge derived from observations." Furthermore, those observations result from the systematic analysis of stated hypotheses via an appropriate experimental apparatus. While the apparatus is not usually the subject of research itself, it is a critical component of any research program.

The apparatuses for experimental research in general purpose distributed database management systems (DDBMSs) have, to date, been limited to simulation and queueing models which address specific problems such as concurrency control [RIES79, GARC78, RAHI79]. While this work has contributed much to the conceptual foundations of DDBMS research, it has also generated many hypotheses (concerning concurrency control, reliability and performance) which have not been subjected to analysis in real system environments. The goal of the DDBMS Project at Honeywell's Corporate Computer Sciences Center (CCSC) is to design and build a "real system" for this purpose. The system, known as DDTS (Distributed Database Testbed System) will provide the apparatus through which hypotheses can be analyzed and quantitative models validated.

From its conception [WEEL79, DEVO80] DDTS has been considered to be a testbed and not a prototype such as SDD-1 [ROTH80]. While the distinction between a testbed and a prototype [BERG80] can at times be small, it is important during design and implementation; the latter emphasizing efficiency and simplicity; the former modularity and flexibility, whenever possible. Even a testbed, however, must be built upon certain architectural foundations (e.g. a single conceptual schema of the database, and the definition of a transaction). Choices must be made in order to build the apparatus. The architecture of DDTS is not intended to be unique (although it is in some areas); rather, readers should notice that we have built upon previous ideas [e.g. KLUG77, STON79, ROTH80, LIND79] as experimental science often does and too frequently computer science does not.

These notes describe, very briefly, our initial "choices." Moreover, we hope they serve to:

- o inform the SIGMOD community about our project
- o generate discussion about experimental research in DDBMS

*This research is partially supported by NSF Grant EC-8007683

- promote similar presentations on other research efforts.

Section 2 presents our view of architectural issues in DDBMS [see also ROTH77]. Section 3 outlines the approach taken in the design of DDTS. We close with some comments in Section 4.

2. DDBMS Issues

This section discusses important issues in the design of DDBMS from two perspectives: first, the Information Architecture describes how information is conceptually modeled, represented, accessed and allocated; and second, the System Architecture addresses processor capability and location, communications, and distributed execution.

2.1 Information Architecture

A. Integrated and Federated Databases: In centralized database management systems, the storage of information in one integrated database is considered essential. In fact, data integration was one of the reasons for initiating database management systems [SIBL76]. Integration was seen as a solution to many problems that existed in earlier file systems, such as data integrity, centralized control, redundancy, and availability. However, integration created problems of security, privacy, and concurrent access to data. These problems are even more severe in DDBMS. Furthermore, some advantages of integration in a centralized system become problems in distributed systems. For example, in a DDBMS where data is replicated to increase reliability and availability, data integrity becomes a problem. In an application where different users want control over their own data, centralized control becomes a problem. This leads to ideas about 'federated' databases [HAMM79] with decentralized control and site autonomy [SELI80]. It is probable that the better approach depends on the user requirements and applications. However, it is not clear whether a 'federated' DDBMS is feasible within a unified framework.

B. Data Models: A multitude of data models have been proposed (see [ELMA80] or [MCLE78] for a survey). However, the vast majority of implemented systems have used only three data models: hierarchical, network, and relational. It is recognized that each of these models has several shortcomings, especially with regard to modelling of information (see [MCLE78] for a discussion). An important issue is whether data models with more general modelling capabilities can be used in database management systems with acceptable performance.

C. Multi-schema/Multi-model Architectures: The importance of data independence from implementation has long been recognized [SIBL76a]. This led to the ANSI/SPARC multi-level architecture for a DBMS, with three different schema levels [KLUG77]. For a DDBMS, even more schema levels may be needed, especially for a system with heterogeneous local DBMS, or with multiple data models for external schemas. Again, an important issue is whether a DDBMS with acceptable performance can be realized using a multi-schema/multi-model architecture.

D. User Interfaces: Many different types of users are expected to use a DBMS; from naive to semi-technical to professional programmers and DBA's. Does the DBMS have to include different types of interfaces for the different classes of users, or is it possible to develop a uniform interface that has sufficient capabilities for all user classes [ROWE81]?

E. Semantic Integrity: In most database applications there are many known constraints that the actual data should obey. The DBMS should, as much as possible, ensure that the data is consistent with the constraints. In most commercial DBMSs, constraint checks are built into the update programs. This approach usually involves reference to actual storage structures, and does not support data independence. In relational DBMSs, general integrity control subsystems [STON74, ESWA75] were proposed. However, their implementations have proven to be quite inefficient. A practical semantic integrity subsystem should allow definition of general constraints, and be able to maintain these contraints during system use and still attain reasonable performance.

F. Heterogeneous Models and Data Translation: An important issue in DDBMSs is the integration of existing local databases into a distributed system. This may require a system where different databases use different data models (hierarchical, network, relational, etc.). An important issue is demonstrating the feasibility of such a system in a unified framework. This will involve algorithms to translate user view data operations to conceptual schema operations, and subsequently to local DBMS operations.

G. Data Allocation: In a DDBMS, allocation of data to distributed locations is an important issue. Tradeoffs exist between replication, with its advantages of quicker data access and greater overall availability, and partitioning, with the advantages of simpler data update algorithms. It is not clear where the optimal compromise is for practical applications. It seems that the degree of replication/partitioning is strongly dependent on the particular application.

2.2 System Architecture

A generally agreed upon system framework for DDBMSs includes a collection of independent processors interconnected via some communication channel used to transfer data and control information [ROTH77]. Beyond this framework a number of issues are of interest in real systems.

A. Processor Distribution: The computers in a DDBMS may be locally distributed (LD) or geographically distributed (GD).

B. Communication: The "channel" between computers may range from low bandwidth packet switched networks (ARPA) to a high bandwidth broadcast system (e.g. Ethernet, satellite, or bus).

C. Separation: The Transaction Management (TM) and Data Management (DM) "functions" [STON79, ROTH80, BERN80] in DDBMSs can be designed with virtual separation (VS) (i.e. they may reside in the same physical processor [ROTH80]) or real separation (RS) (i.e. specialized processors are designed or designated for each function [STON79]).

D. Transaction Definition: All concepts of consistency and recovery are built around this abstraction, generally defined to be a sequence of data manipulation operations which map the database from one consistent state to another [GRAY80].

E. Transaction Processing Model - DDBMSs must provide efficient and reliable concurrent execution of transactions which are defined over an integrated set of logical resources (conceptual and/or external schemas) but are executed over independent processors where physical resources are partitioned and perhaps replicated and where communication is expensive and unreliable. This model is complex and includes at least three problem areas.

- o Optimization - ensuring that costs for intersite communication are reduced and that replication of data and processors is exploited whenever possible [ROTH77, HEVN79, EPST78].

- o Reliable Execution - ensuring that either all of a transaction's effects are seen or none of them are; and once seen they are durable [LIND79, GRAY79]. The problem includes:
  - Resource partitioning: Independent processes must be coordinated, via techniques such as two-phase commit [GRAY78], to guarantee internal consistency.
  - Resource replication: Approaches must be developed which guarantee mutual consistency of replicated objects and yet provide access to objects when only a fraction of those replicas are simultaneously available. [STON79, HAMM80, ALSB76, GARC78]
  - Recovery/restart: Techniques to provide graceful recovery from both transaction and system failures must be available. Proposals for these ideas include the use of private work spaces, shadow files, and logs [BERN80, GRAY79, GRAY80, BLAS79].

- o Concurrency Control - ensuring that the execution history generated by all transactions is equivalent to some serial history. This problem includes two dimensions:
  - Synchronization: An excellent survey of many recently proposed concurrency control methods is found in [BERN80]. All methods

are analyzed as variations of two-phase locking and timestamp ordering.

- Termination: Concurrency control methods may subject transactions to a number of problems such as deadlock, cyclic restart, and starvation [STEA76, KING74] which must be resolved [OBER80, ROSE78, MENA79, BERN80].

3. DDTS Design Choices

We now present our choices for data and system architecture in DDTS. In future versions, alternative designs of different parts of the system will be attempted.

3.1 DDTS Information Architecture

A. Integrated DDBMS: The first version of DDTS will be an integrated, rather than federated DDBMS in the sense that each node will include the same conceptual schema that describes all the data in the system. Subsequent versions may attempt to include only the necessary parts of the global schema at each node: those parts that are necessary to show the users' views, and the actual data stored at the node.

B. Multi-schema Architecture: The ANSI/SPARC 3-schema architecture was generalized to 5-schema levels for a DDBMS [DEVO80] (Figure 1). The external schema is the user interface, and one such schema will exist for each user group. The conceptual schema is a semantic model of the total data content. The global representational schema is a syntactic representation of the total data. The local representational schema is a syntactic description of the data resident at a specific node, while the local internal schema is the local DBMS representation. The data actually stored at each node is described at two levels: the local representational level, which uses the same data model as the global representational level, and the local internal level, which uses the data model of the local DBMS. These levels were chosen as an attempt to modularize and simplify the operational transformations between levels in DDTS.

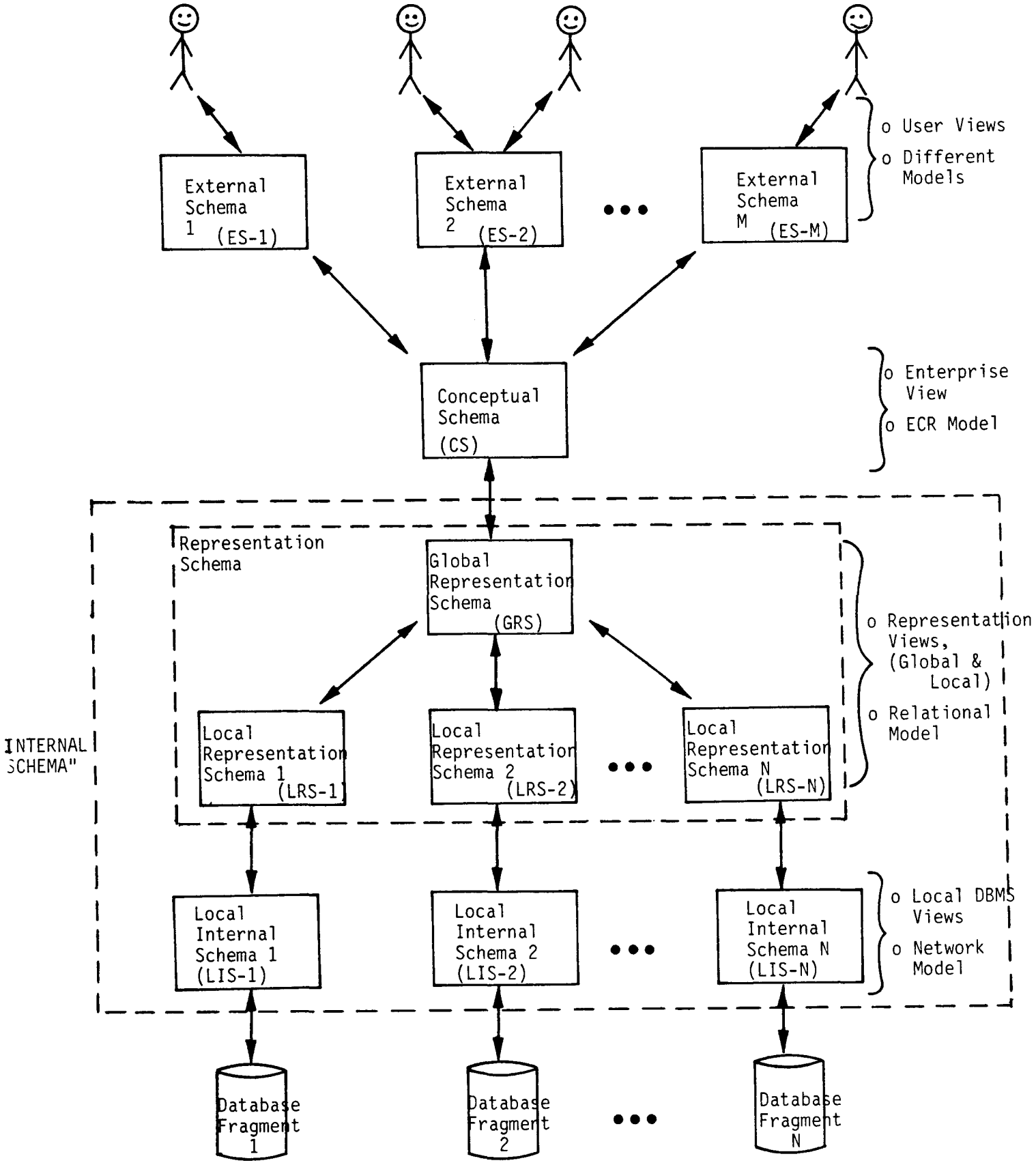

Figure 1: DDTS 5-Schema Information Architecture

C. Data Models: In the first version of DDTS, the conceptual schema will use the Entity-Category-Relationship (ECR) model [WEEL80]. Different external schemas will not be supported. However, subsequent versions will allow the use of different data models for external schemas. The ECR model is a generalization of Chen's Entity-Relationship (ER) model [CHEN76]. The global representation, and the local representation levels will use the relational model [CODD70]. The local DBMS's will be IDS-II systems [HONE78], which use the CODASYL network model. In future versions, we hope to add a relational DBMS (e.g. INGRES [STON76]) as a local DBMS.

D. User Interfaces: The first version of DDTS will include a single high-level user interface. The language GORDAS (Graph Oriented DAta Selection Language) [ELMA81, ELMAsu] will be used as a query and update language. To our knowledge, GORDAS is the first formal language completely defined for the entity-attribute-relationship class of data models, such as the ER/ECR models. In future versions, GORDAS will be expanded to include a programming language for databases. When we use different models at the external schema level, we will import appropriate user interfaces for these models.

E. Semantic Integrity: We have chosen to define semantic integrity constraints during schema definition time. The data definition language for the ECR model includes the capabilities to specify general constraints. Limited capabilities to change constraints will also exist. Update transactions are specified without integrity constraints, and the transaction compiler automatically modifies the transaction to check for the semantic constraints defined in the conceptual schema [ELMA80a]. The transaction is examined as a whole, so that only necessary checks are generated.

F. Data Translation: The user requests, stated in GORDAS, are automatically translated to the relational operators of the representational schema. Another operational translation from the local representational to the local internal DBMS operations is also undertaken. We will have the capabilities to compile transactions so that operational translation is not needlessly repeated.

G. Data Allocation: Recent work by Wong [WONG80] indicates possible qualitative techniques for choice of data partitioning and replication. These techniques emphasize local schema design based on the concepts of "minimal redundancy" and "local sufficiency," which are derived from the semantics of the global schema. We will use these techniques as a starting point for data allocation.

3.2 DDTS System Architecture

A. Processor Distribution: The initial version of DDTS will be implemented on three Honeywell Level 6 minicomputers locally distributed (LD) within a single room.

B. Communication: Communications in DDTS will be provided through a microprocessor based local network [BERG80, KAIN79]. The network currently provides a bus architecture. However, other architectures (e.g. double bus, ring, star) may be realized directly through coax cable reorganization and appropriate software support.

C. Separation: Logically, DDTS consists of a set of Application Processors (APs) and Data Processors (DPs) as shown in Figure 2. Data management is solely the function of DPs, while transaction management is achieved through cooperation among a single AP and required DPs (determined by optimization) for each transaction. The APs and DPs are being designed as "Guardians" [LISK79], abstract processors which support multiple processes with shared memory. Processes in different guardians are designed to communicate via Liskov's "Primitives for Distributed Computing" [LISK79]. Such a design provides virtual separation (VS) of processors. It can be seen that the local distribution (LD)/virtual separation (VS) architecture of DDTS is different from the GD/VS of SDD-1 [ROTH80] and the LD/RS of MUFFIN [STON79].

D. Transaction Definition: The first version of DDTS will define a transaction as any user sequence of GORDAS (see 3.1.D) data manipulation commands defined over the (ECR Model) database conceptual schema. The semantic

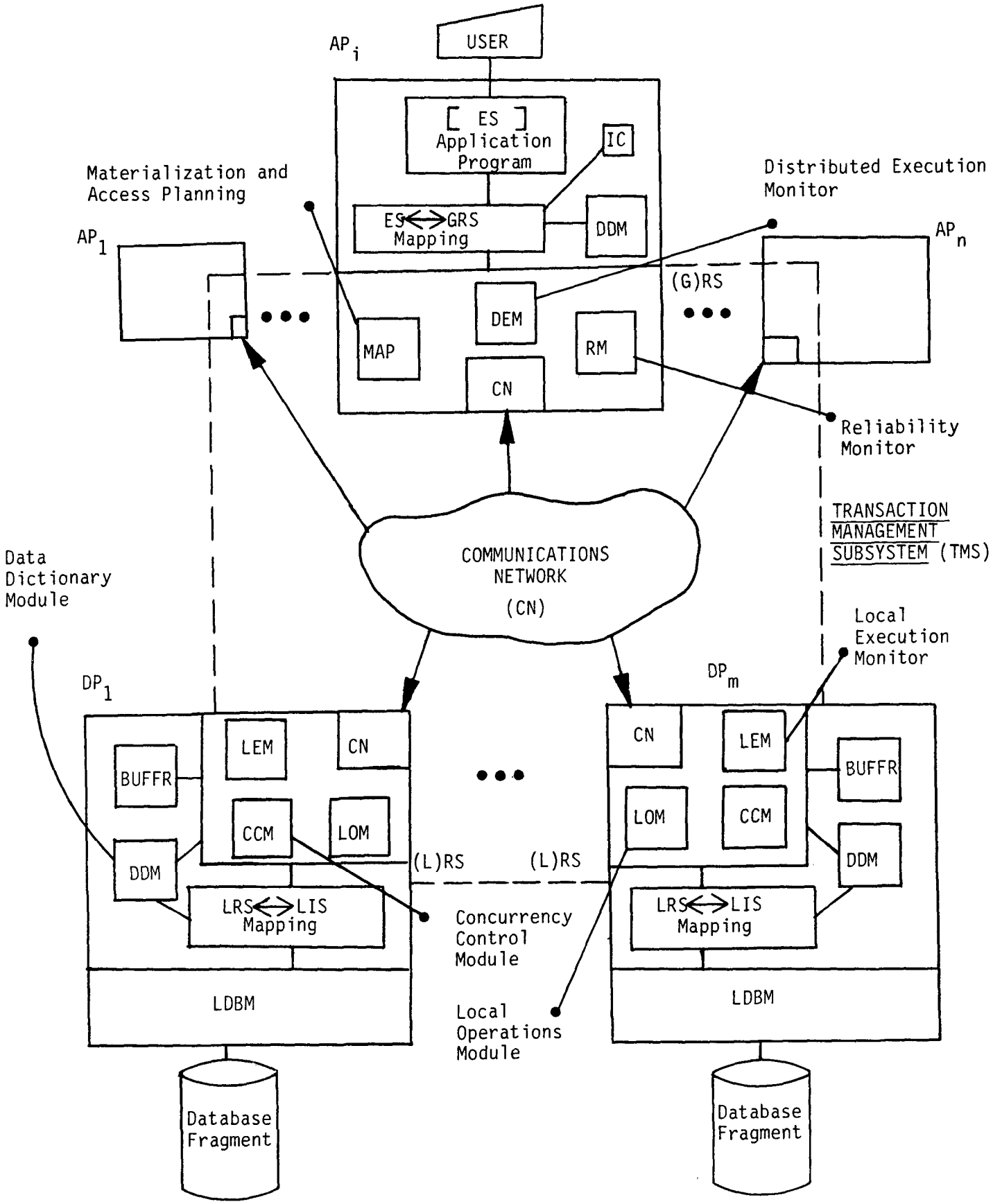

Figure 2: DDTS Functional Components

integrity compiler will automatically insert conditional execution checking into predefined transactions to maintain semantic integrity. Initially, looping will not be supported within transactions.

E. Transaction Processing Model

- Transaction Optimization - Data flow analysis will be used to maximize inter-command parallelism. Additionally, new techniques of dynamic materialization will exploit intra-command parallelism made possible by partitioning and replication of data [HEVN81].

- Reliable Execution - In DDTS, reliable execution is the responsibility of the transaction management subsystem (TMS) shown (in reference to other functional components required to process the 5-Schema Architecture) in Figure 2. A master/slave relationship between a distributed execution monitor (DEM) and required local execution monitors (LEM's) is used to carry out the parallel execution and two-phase commitment of each transaction. The initial version of DDTS will require all data replicas (if any) to be available for updates to occur. While this is a limitation, we are not convinced that proposals in [HAMM80, STON79, GARC78] are realistic for our application. Much of the recovery and restart capability in DDTS will be built upon those facilities existing in the LDBMSs. These include write-ahead logs and backout. Extensions in the areas of commitment logs and the redo-undo concepts of [GRAY80] must be made.

- Concurrency Control - We will experiment with different concurrency control algorithms utilizing synchronization and termination techniques which are compatible with the architecture described.

  For the first phase of DDTS locking facilities and conflict detection mechanism of the Local Data Base Manager, (LDBM) IDS-II [HONE78] will be used. These facilities constitute the synchronization part of the concurrency control. Using these synchronization techniques, we intend to experiment with the following termination techniques.

- A Distributed Deadlock Detection Technique: In which the LDBMs grant the locks "unconditionally" and make the local Wait-For-Graphs (WFG) [OBER80]. When a potential deadlock is noticed a distributed deadlock detection algorithm [OBER80] is activated. The algorithm, then, builds a Global WFG, from which distributed deadlocks are detected. After a deadlock is detected, the DEM determines if rollback is in order prior to two-phase commit processing. We intend to use the rollback facilities of the LDBMs (IDS-II) in DDTS.

- A Distributed Deadlock Prevention Technique: In which locks are granted "conditionally" on the basis of transaction timestamps (age) which prevents deadlocks [ROSE78]. Again, the DEM is responsible for restarting transactions which are selected and have not begun commit processing.

For the second phase of DDTS our goal is to use concurrency control algorithms employing timestamps for synchronization and termination [THOM79, RAHI79, BERN78, REED79]. This means that for this phase we will not be using the conflict detection mechanism and locking facilities of the LDBMs. We have to design and implement synchronization and termination facilities required by concurrency control algorithms employing timestamps.

4. Remarks

We presented our ideas on the important design issues in DDBMS. The DDTS is an experimental tool for quantitatively analyzing some of these issues, and it demonstrates the feasibility of others. We presented our initial design choices for the major components of the system. We hope to keep the SIGMOD readers informed of our progress and results.

We appreciate the comments of Al Hevner (University of Minnesota) and Jim Weeldreyer, who are part of the DDTS design team.

REFERENCES

[ALSB76] Alsberg,P., and Day,J., "A Principle for Resilient Sharing of Distributed Resources," Proc. 2nd Intl. Conf. on Software Engineering, 1976, pp.592-570.

[BERG80] Berg,H.K., "DST - A Distributed System Testbed," Honeywell International Workshop on Database Management Systems Proceedings, Wayzata, Minnesota, October 1980.

[BERN78] Bernstein,P., Rothnie,J., Goodman,N., and Papadimitriou,C., "The Concurrency Control Mechanism of SDD-1: A System for Distributed Databases (The Fully Redundant Case)," IEEE Transactions on Software Engineering, Volume SE-4, Number 3, May 1978, pp. 154-168.

[BERN80] Bernstein,P., and Goodman,N., "Fundamental Algorithms for Concurrency Control in Distributed Database Systems," Computer Corporation of America, Report CCA-80-05, February 1980.

[BLAS79] Blasgen,M., et.al., "System R : An Architectural Update," IBM Report RJ2581, San Jose, California, July 1979.

[CHEN76] Chen,P.P-S., "The Entity-Relationship Model: Towards a Unified View of Data," ACM Transactions on Database Systems, Volume 1, Number 1, March 1976, pp.9-36.

[CODD70] Codd,E.F., "A Relational Model for Large Shared Data Banks," Communications of the ACM, Volume 13, Number 6, June 1970, pp.377-387.

[DENN80] Denning,P., "What is Experimental Computer Science?," Communications of the ACM, Volume 23, Number 10, October 1980, pp.543-544.

[DEVO80] Devor,C., and Weeldreyer,J., "DDTS: A Testbed for Distributed Database Research," Proc. ACM Pacific '80 Conference, San Francisco, California, November 1980, pp. 86-94.

[ELMA80] El-Masri,R., "On the Design, Use, and Integration of Data Models," Ph.D. Thesis, Computer Science Department, Stanford University, Report STAN-CS-80-801, May 1980.

[ELMA80a] Elmasri,R., "Semantic Integrity in DDTS (Distributed Database Testbed System)," Honeywell CCSC, Technical Report HR-80-274, Bloomington, Minnesota, December 1980.

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

[ELMAsu] Elmasri,R., "GORDAS: A Data Selection Language for the Entity-Relationship Model" (submitted for publication).

[EPST78] Epstein,R., Stonebraker,M., and Wong.,E., "Distributed Query Processing in a Relational Data Base System," Proceedings of the ACM SIGMOD 1978 Conference, Austin, Texas, May 1978, pp.169-180.

[ESWA75] Eswaran,K., and Chamberlin,D., "Functional Specifications of a Subsystem for Database Integrity," VLDB 75 Proceedings, Framingham, Massachusetts, September 1975, pp.48-68.

[GARC78] Garcia-Molina,H., "Performance Comparison of Two Update Algorithms for Distributed Databases," Proc. 3rd Berkeley Conference on Distributed Data Management, San Francisco, California, August 1978, pp. 108-119.

[GRAY78] Gray.J., "Notes on Data Base Operating Systems," Lecture Notes in Computer Science, Volume 60, Springer-Verlag, New York, 1978, pp.393-481.

[GRAY79] Gray,J., et.al., "The Recovery Manager of a Data Management System," IBM Report 2623, San Jose, California, August 1979.

[GRAY80] Gray,J., "A Transaction Model," IBM Research Report, San Jose, California, February 1980.

[HAMM79] Hammer,M., and McLeod,D., "On Database Management System Architecture," MIT, Report MIT/LCS/TM-141, October 1979.

[HAMM80] Hammer,M., and Shipman,D., "Reliability Mechanisms for SDD-1: A System for Distributed Databases," ACM Transactions on Database Systems, Volume 5, Number 4, December 1980, pp. 431-466.

[HEVN79] Hevner,A., and Yao,S., "Query Processing in a Distributed Databases," IEEE Transactions on Software Engineering, Volume SE-5, Number 3, May 1979, pp. 177-187.

[HEVN81] Hevner,A., "Transaction Optimization in Distributed Database Systems," (in preparation).

[HONE78] Honeywell Information Systems, GCOS6 IDS/II User's Guide, Waltham, Massachusetts, November 1978.

[KAIN79] Kain,R., Franta,W., and Jelatis,G., "CHIMPNET: A Networking Testbed," Proc. 4th Conf. on Local Computer Networks, October 1979, pp. 27-36.

[KING74] King,P., and Collmeyer,A., "Database Sharing - An Efficient Mechanism for Supporting Concurrent Processes," Proceedings of the National Computer Conference, AFIPS, Volume 43, May 1974.

[KLUG77] Klug,A., and Tsichritzis,D., (eds.), The ANSI/X3/SPARC DBMS Framework Report of the Study Group on Database Management Systems, AFIPS Press, 1977.

[LIND79] Lindsay,B., et.al., "Notes on Distributed Databases," IBM Research Report RJ2571, San Jose, California, 1979.

[LISK79] Liskov,B., "Primitives for Distributed Computing," MIT/LCS Technical Note, August 1979.

[MCLE78] McLeod,D., "A Semantic Data Base Model and its Associated Structured User Interface," MIT Technical Report MIT/LCS/TR-214, Cambridge, Massachusetts, August 1978.

[MENA79] Menasce,D., and Muntz,R., "Locking and Deadlock Detection in Distributed Data Bases," IEEE Transactions on Software Engineering, Volume SE-5, Number 3, May 1979, pp. 195-222.

[OBER80] Obermarck,R., "Global Deadlock Detection Algorithm," IBM Report RJ2845, San Jose, California, June 1980.

[RAHI79] Rahimi,S., and Franta,W., "A Posted Update Approach to Concurrency Control in Distributed Database Systems," Proc. 1st International Conference on Distributed Systems, Huntsville, Alabama, October 1979, pp. 632-641.

[REED79] Reed,D., and Kanodia,R., "Synchronization with Event Counts and Sequences," Communications of the ACM, Volume 22, Number 2, February 1979, pp. 115-123.

[RIES79] Ries,D., "The Effects of Concurrency Control on the Performance of a Distributed Data Management System," Proc. 4th Berkeley Conference on Distributed Data Management, August 1979, pp. 75-112.

[ROSE78] Rosenkrantz,D., Stearns,R., and Lewis,R., "System Level Concurrency Control for Distributed Database Systems," ACM Transactions on Database Systems, Volume 3, Number 2, June 1978, pp. 178-198.

[ROTH77] Rothnie,J., and Goodman,N., "A Survey of Research and Develoment in Distributed Database Management," Proceedings of the Third VLDB Conference, Tokyo, Japan, October 1977, pp. 48-62.

[ROTH80] Rothnie,J., et.al., "Introduction to a System for Distributed Databases (SDD-1)," ACM Transactions on Database Systems, Volume 5, Number 1, March 1980, pp.1-17.

[ROWE81] Rowe,L., and Stonebraker,M., "Architecture of Future Database Systems," SIGMOD Record, Volume 11, Number 1, January 1981, pp.30-45.

[SELI80] Selinger,P., Lindsay,B., Obermarck,R., and Yost,B., "R*: A Distributed Relational Database Management System," IBM, San Jose California, Notes presented at VLDB Conf., Montreal, Canada 1979.

[SIBL76] Sibley,E.H., "The Development of Data-Base Technology," ACM Computing Surveys, Volume 8, Number 1, March 1976, pp.1-6.

[SIBL76a] Sibley,E.H., and Fry,J., "Evolution of Data-Base Management Systems," ACM Computing Surveys, Volume 8, Number 1, March 1976, pp.7-42.

[STEA76] Stearns,R., Lewis,P., and Rosenkrantz,D., "Concurrency Controls for Database Systems," Proceedings of the 17th Annual Symposium on Foundations of Computer Science, Houston, Texas, October 1976, pp.19-32.

[STON74] Stonebraker,M., "High Level Integrity Assurance in a Relational Data Base Management System," UC Berkeley, Technical Report ERL-M473, Berkeley, California, May 1974.

[STON76] Stonebraker,M., Wong,E., and Kreps,P., "The Design and Implementation of INGRES," ACM Transactions on Database Systems, Volume 1, Number 3, September 1976, pp.189-222.

[STON79] Stonebraker,M., "MUFFIN: A Distributed Database Machine," Report UCB/ERL-M79/28, Electronics Research Lab., University of California, Berkeley, California, 1979.

[THOM79] Thomas,R., "A Majority Consensus Approach to Concurrency Control for Multiple Copy Database," ACM Transactions on Database Systems, Volume 4, Number 2, June 1979, pp.180-209.

[WEEL79] Weeldreyer,J., "Preliminary Design for a Distributed Database Test System," Honeywell CCSC, Technical Report HR-80-253, Bloomington, Minnesota, June 1979.

[WEEL80] Weeldreyer,J.A., "Structural Aspects of the Entity-CategoryRelationship Model of Data," Honeywell CCSC, Technical Report, HR-80-251, Bloomington, Minnesota, March 1980.

[WONG80] Wong,E., "The Design of Representation Schemas," Honeywell CCSC Technical Report HR-80-265, Bloomington, Minnesota, July 1980.