1991

Interoperability in Multidatabase Systems

Omran A. Bukhres

Ahmed K. Elmagarmid
Purdue University, ake@cs.purdue.edu

Report Number:
91-021

INTEROPERABILITY IN MULTIDATABASE SYSTEMS

Omran A. Bukhres
Ahmed K. Elmagarmid

CSD-TR-91-021
March 1991

INTEROPERABILITY IN MULTIDATABASE SYSTEMS

Omran A. Bukhres
Computer Science Department
Moorhead State University
Moorhead, MN. 56563

Ahmed K. Elmagarmid
Computer Sciences Department
Purdue University
West Lafayette, IN. 47907

February 28, 1991

To appear in the Encyclopedia of Microcomputers
October 1991, Volume 9

INTRODUCTION

1.1 Overview of the Problem

Organization-wide access to data and software resources requires the interconnection of previously isolated applications and systems. An end-user in a heterogeneous computing environment should be able not only to invoke multiple existing application systems but also to coordinate their interactions and associations. In today's computing world, databases are proposed as a solution to the problem of shared access to heterogeneous resources, both software and hardware, that exist in multiple autonomous applications.

Organizations typically contain a collection of different departments, each making autonomous decisions about how it is to operate within the larger enterprize. This article includes definition of the autonomous design of databases in support of individual information systems. Several technical issues need to be addressed in order to fully describe autonomous database systems.

Because the article is intended to describe the current state-of-the-art in the heterogeneous database area, the reader

should be familiar with fundamental database management issues as well as with a number of different data models such as relational, entity-relationship, network and functional data models. For additional information about these basic ideas, see [31] for a review.

1.2 Heterogeneous Databases

"Heterogeneous databases," "multidatabases," and "federated databases" are terms often used interchangeably. These terms must be precisely defined for the purpose of this discussion.

A distributed database management system (distributed DBMS) is one capable of managing a collection of databases distributed over a network at different geographical sites. Each site in the distributed DBMS can have its own CPU-DBMS, database administrator, terminals, users, local storage, data communication manager, and local autonomy.

One of the objectives of distributed DBMS is location transparency, in which the user of the database is not required to know which site has the integrated data. This transparency simplifies the logic of programs and allows data movement as usage patterns change. A second objective is the transparency of data fragmentation, in which a logical object

(e.g., a file) can be divided into multiple physical pieces for storage purposes and can be accessed at different sites. A third objective is data replication and its transparency, in which a logical object has more than one physical copy at different sites; this situation has the advantage of resource availability, but the disadvantage of update overhead. The user need not know that the data is replicated.

With the transparency of location, data fragmentation, and data replication, the user need not know that the database system is distributed. Local autonomy is an essential element of distributed DBMS design. The system is distributed along lines of the logical and physical distribution of the enterprise, and allows local control over local data, a situation which means local accountability and dependency on a remote data processing center.

Distributed database systems provide a high degree of concurrency and resource sharing, but are not trouble-free. Concurrency control strategies, global deadlocks, recovery, and reliability are some of the problems associated with a distributed computing environment.

A multidatabase system (MDBS) allows operations on multiple databases. A multidatabase management system (MDBMS) is the software that provides integrated management of the

interactions among theses systems. A multidatabase environment makes assumptions about the *heterogeneity* and *homogeneity* of the individual DBMSs, and whether or not these autonomous databases are distributed. It is therefore possible to have two identical copies of a DBMS autonomously managing two databases that may have been developed for different purposes [28].

A *distributed multidatabase system* (D-MDBS) is one in which the individual database and DBMSs are located on multiple processors. A D-MDBS is not the same as a distributed database system. In the former, the database management software run at each site is tightly integrated and is fully aware of the existence of the other databases and their associated software at other sites. In the latter, the autonomy of each database is the essential consideration.

A heterogeneous database system (HDBS) is a multidatabase system in which the data model used to structure the data in the autonomous databases is different in every local database. A heterogeneous database management environment extends the possibilities of heterogeneity to the management algorithms. Thus a heterogeneous database environment may exist if, for example, different transaction management strategies are used. Heterogeneity can exist in architectures, data models, schemas, query languages and optimization techniques and in

transaction management. Each of these forms of heterogeneity presents unique problems.

A (HDBMS) is a system created to provide operations on databases managed by autonomous and heterogeneous database systems. For many applications, a HDBMS is preferred to a simple MDBMS approach.

1.3 Heterogeneity in Database Systems

A Database Management System (DBMS) is a software that manages a collection of structured data. Management includes providing data management services including data access, constraint enforcement, and consistency management. Heterogeneity can exist in architectures, data models, schemas, query languages and optimization techniques and in transaction management. Each of these forms of heterogeneity presents unique problems.

For example, in the architecture of databases heterogeneity can exist at any stage. ANSI/SPARC [36] and (see Figure 1) define a three level schema reference model which is the starting point for a discussion of the major issues in heterogeneous databases.

The three-schema architecture provides for different

definitions at different levels within the database. Conceptual schemas define the data as a whole. That is, all the data used by the database, or accessible by the database management system, is described conceptually at this level. Individual users are typically interested in some subset (possibly complete) of the data described by the conceptual schema. Externals schema of external views (or simply views) describe the portion of the data of interest to an individual user. Physical representation of the data is defined with an internal schema that describes how the conceptual schema can be mapped to the data stored in the database per se. Unfortunately this architectural model cannot be immediately applied to heterogeneous database management systems, since it is missing a component representing the integration of multiple user views defined in different languages.

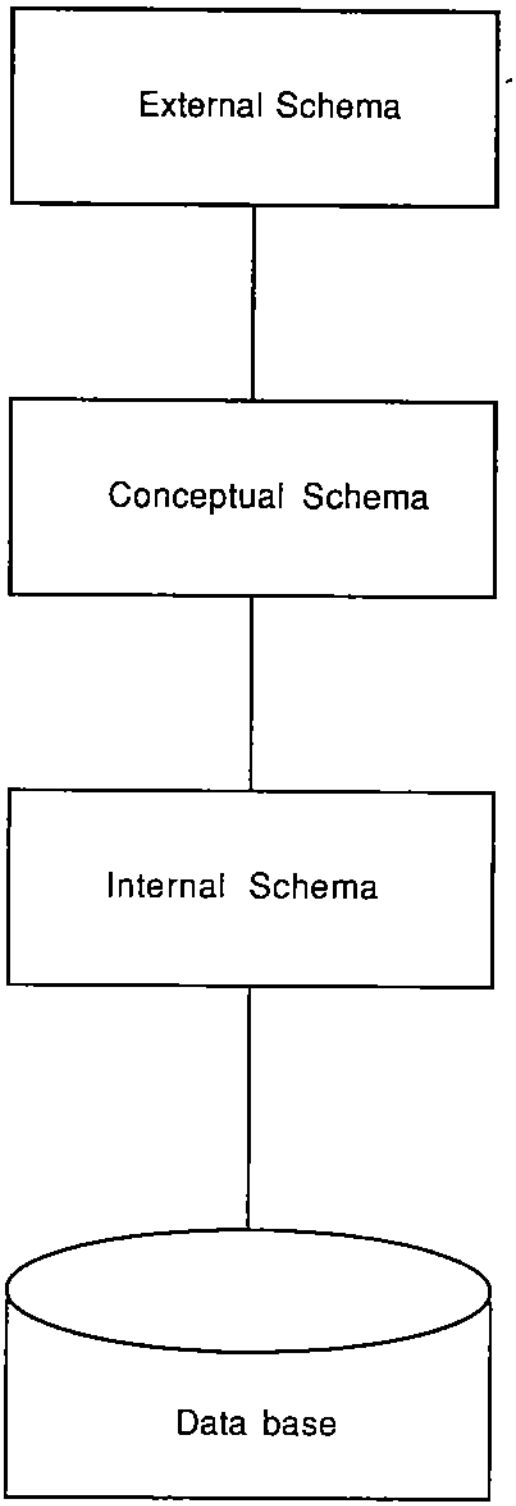

Figure 1: ANSI/SPARC ARCHITECTURE [36]

1.4 Data Models

Those data models available to database designers which were described by Tsichritzis and Lochovsky [37] emphasize the rich variety of data which can be used at any of the conceptual, external or internal schema levels. Here once again the heterogeneity may exist at all levels; however, the problem in this case is more complicated than in the architecture level: no two models are the same, because each depends on the database designer who has his own view of the world.

Tamer Ozsu [28] defined this problem into possible scenarios:

Inter-model discrepancy: one which occurs as a result of using two different data models for the same real-world situation.

Intra-model discrepancy: one which occurs because two designers interpret the real world differently but apply the same data model.

Resolving these types of variances is one of the major challenges for heterogeneous database systems designers.

Schemas and Heterogeneity

2.1 Schemas

In database systems there exist three types of schemas. A Schema, a Subschema, or a View is a representation of the structure (or syntax), semantics, and constraints on the use of a database. A schema is thus a collection of schema objects, and a subschema is a collection of subsets of that schema's objects.

Figure 2 [28] characterizes the layers of a schema composed of four views that work very well in a distributed database system environment which prohibits any autonomy. Each conceptual schema of the local database is integrated to form a global conceptual schema depicting the data found across the entire distributed system.

A brief description of the three different layers of schemas:

-Internal Schema:

Conceptual schemas are essential; they are the first level in any architecture with which other database systems need to be made harmonious. Figure 2 illustrates an architecture in which each local internal schema is managed by one local conceptual schema, just as it would have been in the stand-

alone database situation. The local conceptual schema effectively isolates the global system from the details of management.

-Conceptual Schema:

The conceptual schema is the plate that describes the structure of the whole database for all users. It is a global description of the database that does not reveal the detail of physical storage, entities, data types, and their relationships. Although numerous representation formalism for the conceptual schema have been proposed, only a few are widely available commercially and accepted: entity-relationship, network, hierarchical, and relational. Although numerous distributed database management systems (D-DBMS) have been proposed, in fact all are actually relational.

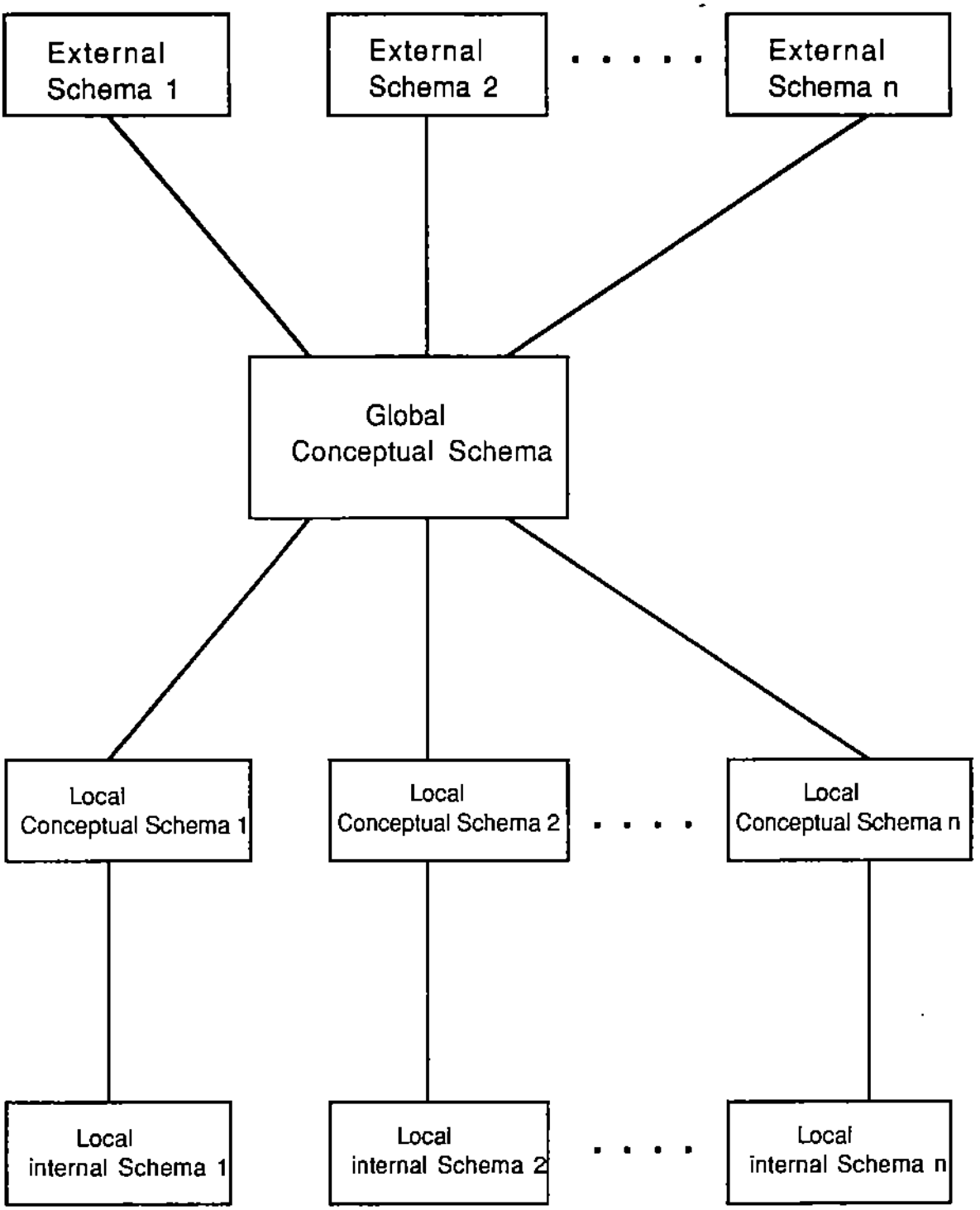

Figure 2: Four-Level Schema Architecture [28]

-External Schema:

Location transparency and replication transparency are two important elements in distributed database design. Heterogeneous database management systems must provide an additional level of transparency, known as data model transparency [28].

Future research should concentrate on a system that would have the following criteria. First, the existence of data model transparency is essential for distributed database systems. Second, the information stored in the database systems must be transparent to the general user (see Figure 2). The global conceptual schema is responsible for mapping the external views and the data model.

2.2 Homogeneous Approach

The global conceptual schema is the basis of the architecture of the homogeneous distributed database systems. One of the important characteristics of a homogeneous approach is that access to global data is provided through the use of global external schemas which have been defined on the global conceptual schema. Moreover, the global user is certainly different from the local user, who utilizes both a different data model and possibly a different data language.

Users accessing global data are affected by this schema's integrations. Maintaining local autonomy is also very important in the use of a global conceptual schema. In the homogeneous approach, the local databases are left untouched, and local users continue to function as they had done before the integration; they are also unaware that their database in part of a global one.

Problems found in heterogeneous databases are best approached by schema integration. Much research has been done in attempt to produce a canonical data representation [28]. One advantage of this approach is that there is no need to translate from global conceptual schema to the single local conceptual schema.

In most of the cases, global external schemas are defined in such a way that the global database can be considered as an ANSI/SPARC database.

2.3 Heterogeneous Approach

In the heterogeneous approach, the user is allowed to access global data by using the external schema which has been defined in local database language. In the presence of global external schema, the user is made to use different data models when both local and global databases are accessed.

In Figure 3 the global conceptual schema is defined as the integration of all the local conceptual schemas. Only small parts of the local databases are accessed by global users. Cardenas [3] is one of the few papers that have considered the multi-language approach. Most of the time, the global external schema and the global conceptual schema are defined with the same data model and language.

2.4 Federated Databases

As mentioned, a federated database system provides operations on databases managed by autonomous, and possibly heterogeneous, database systems. Both the databases and database management systems perform crucial functions in federated databases. While local databases are autonomous in the system, they still participate in the global processing or, in other words, the federation.

In federated database systems, there are a number of schemas to facilitate the absolute autonomy that exists in these type of databases. For example, an import schema is basically the characterization of the data being imported for the database users from another database. An export schema is the description of the subset being exported to other database(s). A federated schema is actually an integration of multiple

export schemas, which describes the data distribution over the federated database. While making data available to all users, the federated databases provide a great deal of security to the local database systems.

Hiembigner and McLeod [19], Rusinkiewicz [32], Litwin [26], Rajinikanth [29], and Smith [33] are excellent references for more detailed information.

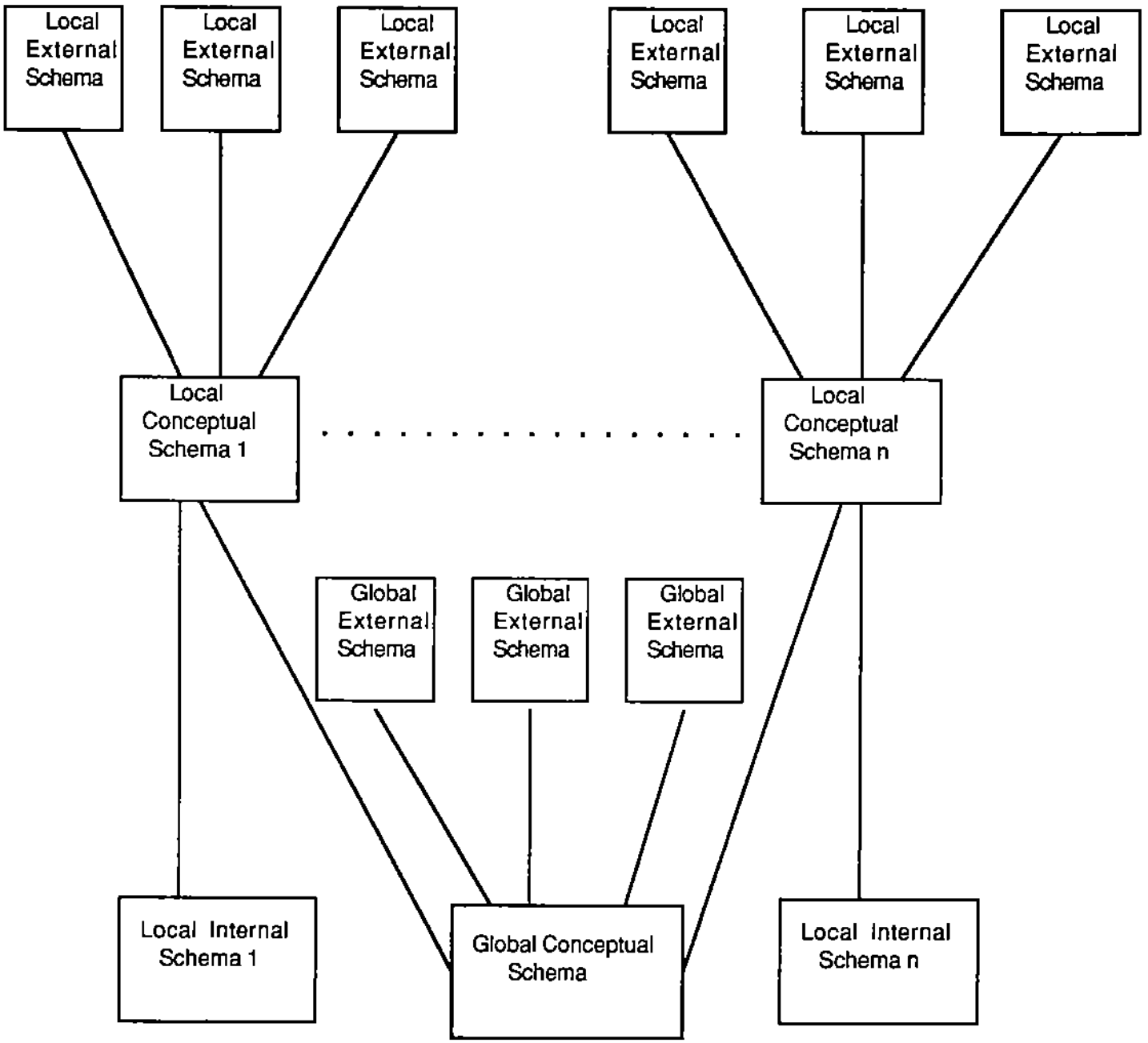

Figure 3: Global Conceptual Schema [28]

Schema Integration and Translation

3.1 Schema Translation

In a heterogeneous database, combining the integration step and the transaction at one time provides the integrator (software that integrates the different schemas) with all the information regarding the entire global database. Obviously, the integrator can make the tradeoff determination as to which local external schema has precedence in a conflict. It would be an unreasonable expectation with this approach to expect the integrator to be aware of tradeoffs done through several schemas.

The heterogeneous database characteristic problem is not changed by the transaction/integration steps. Some heterogeneous systems, by assuming that each is a global external schema, define the transaction step from a local conceptual schema; this approach converts the difficulty into a new integration problem. The definition of the global conceptual schema is then integrated by the global external schema, which is basically a number of local conceptual schemas.

The goal of changing easily from one type of DBMS to another has been the direction of recent research regarding translation of definitions from one schema to another. The concern here is the phase verification of the translation procedure, rather than the specifics. The automation of the data model integration process can be accomplished through modularization of the task, using two distinct models. First, each schema is regarded as being distinctive, recognized and globally accessible to all local schemas. The requirement that each relation be distinctively identified and that all attributes known is a universal relation assumption often applied to relational database schemas.

3.2 Relational Schema

A few years ago there was a trend to the translation from the network model to the relational one. The translation problem, from source schema to relational one, has two parts, the translation phase and the restructuring point [21, 22, 23, 37, 39]. Research work also had been done in the area of interpretation simulations which produce a relational interface for some source data models. The relational appearance of the database is very helpful for the general user of the database. This latter strategy demands that the translation be accomplished at run-time for ad hoc queries, although the need for database restructuring and translation

was obviated. The federated database strategy [19] is promoted by the translation definition as an interface problem, and the heterogeneous approach described in an earlier section.

Many technicalities, but very little universality immediately relevant to other mapping, have been contributed by the specific translators produced by the one-to-one data model mapping. Serious penetration into the common standards is given by the classification of major components to be mapped.

3.3 Schema Integration

The term "view integration" is frequently assigned to the operation of integrating a number of local external schemas, basically an alleviation of schema integration, as in the homogeneous approach. So that the specifications of each user can be adjusted, the view integration is accomplished during the database planning.

One good characteristic of a global conceptual schema which contains an appropriate data model is that it furnishes a satisfactory summary of the data and a specific depiction of any limitations forced upon the data. Most of the time, however, almost any data model can be chosen to portray the

global conceptual schema.

One condition resulting after translations is integration in the heterogeneous database systems so that commonalities taking part in the transitional canonical schemas can be recognized. Repeated representation of the same data item in the global conceptual schema should not be allowed. The procedures for consolidating a number of isolated transitional schemas will be addressed in later sections, and an explanation of the process involved in the integration achievements be provided.

3.4 Possible Schema Integration

Three basic types of components exist within the entity relationship approach: entities, relationships, and attributes. The collection of such entities, relationships, and attributes that exists in the model is the universe of discourse, elements of which may be represented in the schema in different ways. The identification of which element in the

schema represents the same element in the universe of discourse is necessary so that each element exists uniquely in the integrated schema. For equivalence, it is necessary that all components of one schema be equivalently represented in the other, since equivalence between schemas can be defined

in terms of components. Great latitude in what constitutes equivalence in database schemas is provided by this definition, which should give the integration the maximum discretion in selecting the most natural representation for a given application. Schema integration, therefore, becomes a problem of identifying the components of a database which are equivalent, best representation selection for the integrated schema and, finally the inclusion of those non-overlapping components of each database.

Transaction Management

4.1 Transaction Processing in Multi-site DBMS

A distributed database system, as defined previously, is a collection of centralized databases connected by a computer network. In the case of distributed DBMS, the model is different from the centralized one. In the distributed model [1] Figure 4, the data does not reside in one single location, but in different sites in the network. Each site is a centralized database system which stores a portion of the database. Each data object is stored exactly on one site so there is no data replication.

It is assumed that a global transaction issues each of its operations to the closest site in the network. It is the responsibility of the global transaction manager (GTM) to manage the execution of the transactions that execute at several sites. The GTM will communicate with local transaction managers to coordinate the submission of the transaction's operations to the different schedulers at other sites which have the data needed to process the operation. A more detailed description of this logical model and the manner in which it is used as a basis for the concurrency control theory can be found in [1].

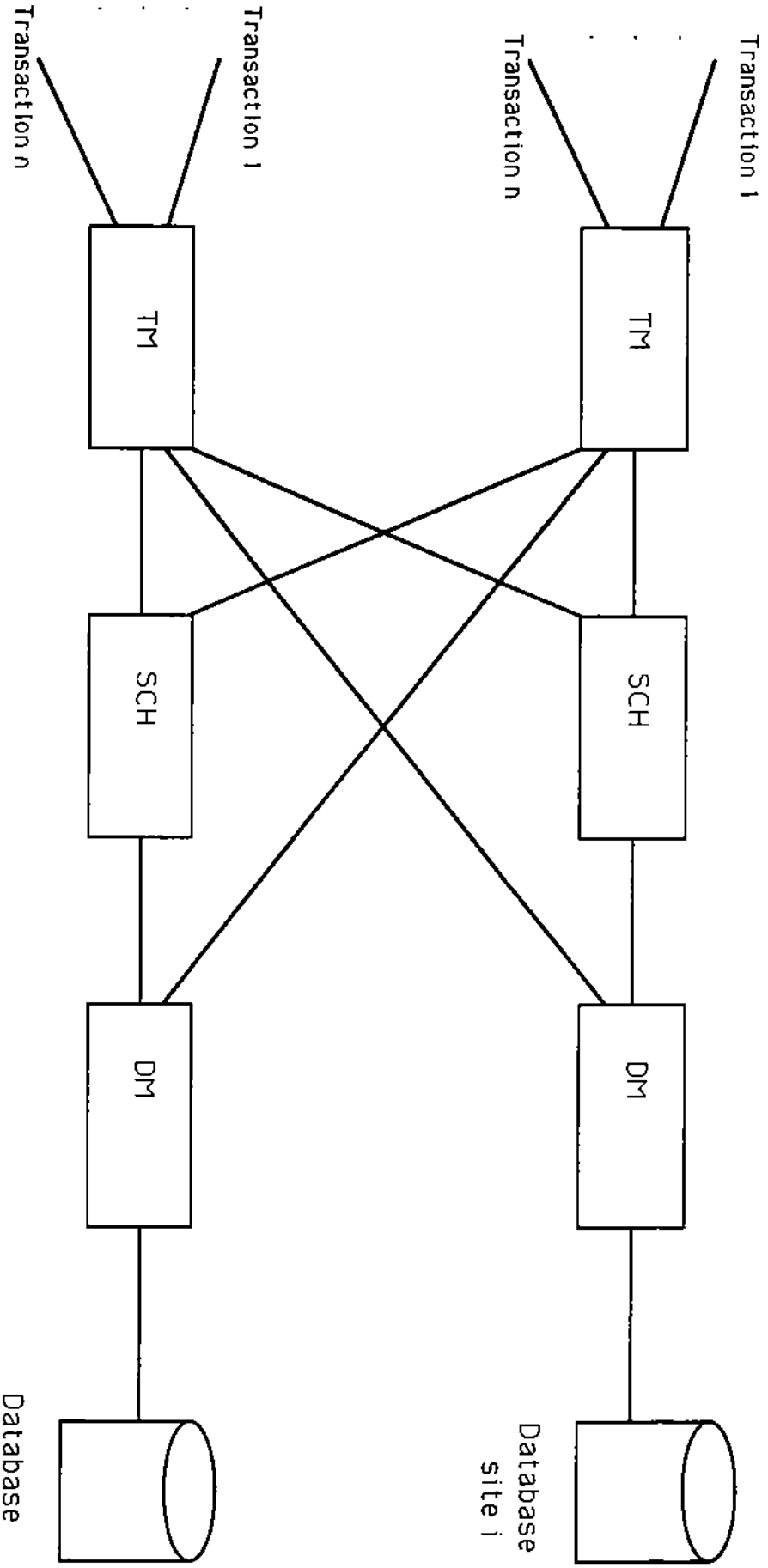

Figure 4: Logical Model for Multi-Site DBMS

4.2 Transaction Properties

A transaction is an atomic unit of database work which is composed of operations such as read, write, abort, and commit. Informally, a transaction is an execution of a program that accesses a shared database. An example of a simple transaction is shown in Figure 5. Transaction T1 in Figure 5 is composed of four database operations--start, read, write, and commit. Transaction T1 increments a data item X by one.

```
Transaction T1
    Begin
        Start;
        temp = Read (X);
        temp = temp + 1;
        write (X,temp);
        commit;
    End
```

Figure 5: Transaction

Transactions in distributed DBMS have three characteristics or properties. The first is atomicity. A transaction must be executed until committed; if the transaction is aborted, there must be no indication that the transaction was in the database system. A transaction T is atomic, if all or none of its operations is performed.

The second characteristic is consistency. A transaction T must take the database state from one consistent state to another.

The third characteristic of transactions in distributed DBMS is durability; once a transaction is committed, the DBMS must guarantee that its results are permanent.

There are other characteristics that relate to the datbase system rather than the transactions. Some researchers argue that isolation and serailizabilty are not transaction's property rather they are characteristics which are enforced by the transaction manger [11, 12, 13, 22]. The first systems' characteristic is isolation, which gives the user the illusion of being the sole user of the system. In other words, incomplete transactions cannot reveal results to others until they commit, in order to avoid cascading aborts.

The second systems' characteristic is serializability. A transaction T is said to be serializable if its effects are equivalent to some serial order of execution. In most distributed DBMS, serializability is used as a correctness criterion. An execution is serializable if it produces the same output and has the same effect on the database as some serial execution of the same transaction. Since serial

executions are correct, and since each serializable execution has the same effect as serial execution, serializable executions are also correct.

4.3 Transaction Management in Heterogeneous Environment

As mentioned, the autonomy of local databases is an important characteristic of heterogeneous database systems. There are two kinds of transactions in a heterogeneous database, a local transaction and a global one. The former is a transaction that accesses data item(s) at a local data base, the latter is a transaction that accesses data item(s) in more than one local database. A global transaction is thus a set of subtransactions. In order for the database to be in a consistent state, the executions of both the local and global transaction have to be correct. Moreover, both local and global transactions must convert the local and global databases from one consistent state to another consistent state.

The transaction management model is depicted in Figure 6 [11, 12, 14 16]. The transaction model consists of a global transaction manager (GTM), a global data manager, and a set of local database systems. As shown in the Figure 6, once a global transaction is submitted to the heterogeneous database

system, it is broken down into subtransactions by the GDM. Consequently, the GTM yields these subtransactions to the appropriate local database systems. It is the responsibility of the GTM to make sure that the state of the global database is continuously consistent.

Another controversial issue in heterogeneous database systems is the heterogeneity among the transaction managers. At the time of this writing, there is no known research that leads

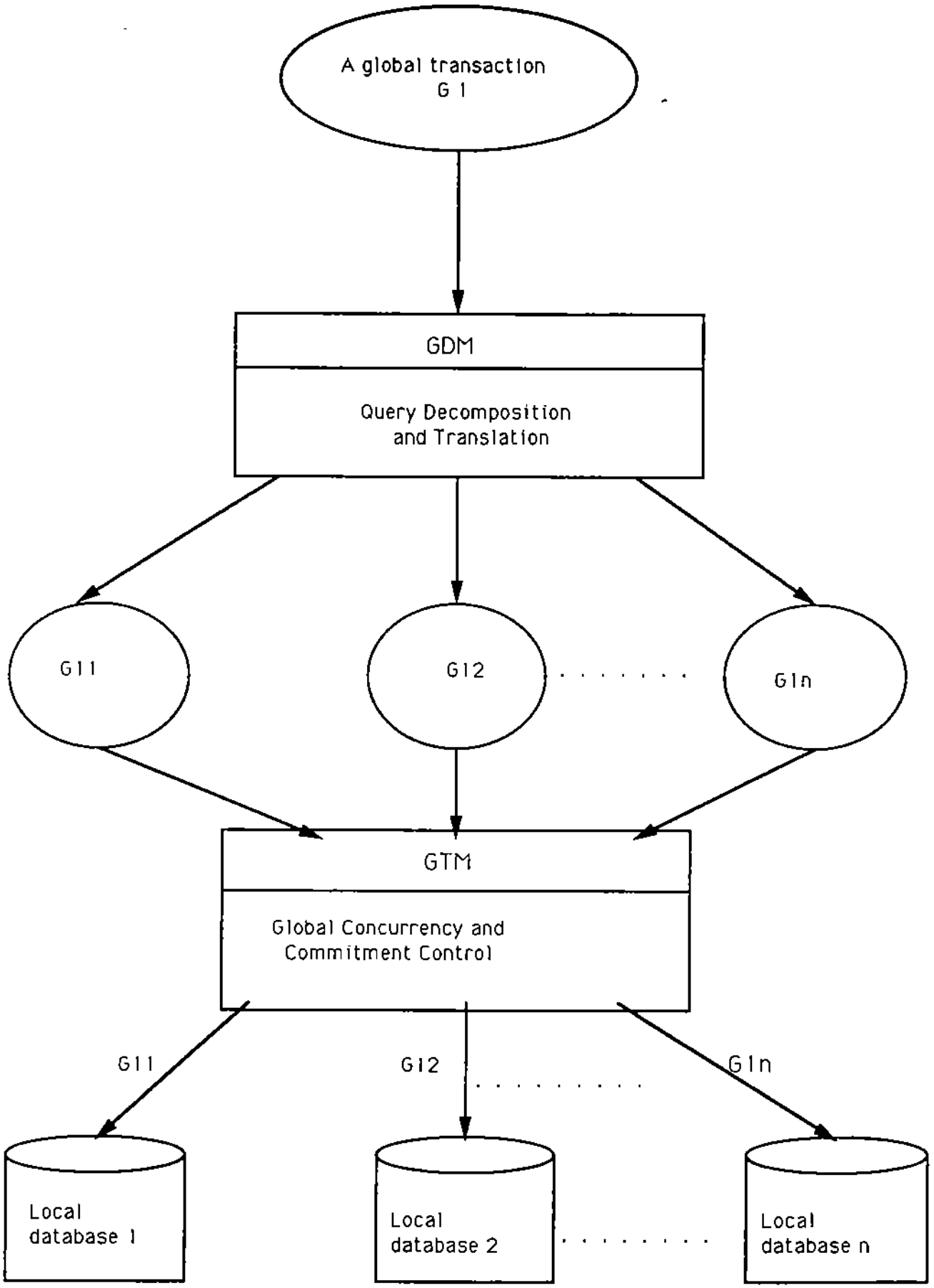

Figure 6: Transaction Processing Model [12]

to sound solutions to this complex problem. However, some studies have investigated this problem in detail. If autonomy is broken there is a hazard that local databases are being altered, a reduction of the problem to the one found in homogeneous distributed databases. The first problem needs to be understood before headway can be made on the second. If there is a discovery related to the autonomy problems, the heterogeneous issues may then become evident.

Heterogeneous database updates may be possible without a mechanism to guarantee transaction management. For example, off-line updates might occur in which transactions are submitted so that no integrity violations occur; essentially, the only user of the global database would be the global transaction manager, who would be responsible for proper updates [14, 15, 16]. This approach would abolish any local transaction concurrency, but would furnish essential revision while guaranteeing local database autonomy. There is still promise of a breakthrough of a more general on-line transaction management model.

4.4 Autonomy

As mentioned earlier, autonomy of the local databases is an important characteristic of heterogeneous database systems. Heterogeneity presupposes autonomy because there are two distinct managers with distinct authority procedures. To furnish an integrated strategy, analogous situations should include solutions to heterogeneous transaction management problems.

Global transactions submitted to local databases should have the same order of execution in order to be serializable. Autonomous databases make their own transaction-ordering decisions, and therefore the global manager is unable to guarantee the transaction ordering.

Elmagarmid and Du [17] propose an algorithm that also sacrifices some local autonomy, but permits an increase in concurrency.

The two important issues in heterogeneous environment are autonomy and heterogeneity. These two matters could overrun each other. To enable both of these issues, or to disable either one of them, would require a mixture of different problems. Primarily, for homogeneous systems with no autonomy, most of the questions are answered and some of them

are applied. In the case where the systems have non-autonomous transaction managers, the transaction processing can be regulated and their execution order could be altered. However, when the system has the same autonomous transaction managers, the control of the transaction processing has a high degree of complexity. The fact that the global transaction manager is not allowed to interfere with the local transaction managers is the center of the problem. The highest degree of complexity occurs when the global transaction managers are different and autonomous. At the time of this writing, there is no solution to this specific problem. This is a real challenge for the researchers in heterogeneous systems.

Query Optimization

5.1 Query Formation

With the already defined heterogeneous environment, it is very attractive to support queries that need to combine related data from the several local databases. Since these databases might have been designed independently of each other and may operate autonomously, some incompatibilities among them are possible. For example, the databases may have different user-interface languages, may have differing domain definitions for attributes whose values should be comparable, and may have only partially compatible entity descriptors even when the same type of entity is represented.

A possible approach to the problem of processing user queries in such environments is to use a global schema created by integrating local site schemas. Queries issued with reference to the global schema can be broken down into subqueries for each database, and the data obtained from executing subqueries can be combined into a final result. Issues of schema integration have been discussed in [7, 27]. Query decomposition, translation, and evaluation in a federated database environment have been addressed in [2, 7, 8, 9, 18]. The main advantage of this approach is that it provides a high

level of location and distribution transparency.

Another possible solution is to provide a facility allowing the user to derive an answer to a multidatabase query by explicitly manipulating several databases [25, 26]. The main advantage of this approach is that it does not require explicit global schema integration, which is frequently difficult and sometimes undesirable. Instead, the appropriate sub-schema (external view) can be dynamically derived by a user posing a query [26].

This section introduces an approach to schema integration and query formulation that is based on graphical manipulation of individual databases. Schema integration may be done through a process similar to view definition, and can be regarded as a step in the process of formulating a query requiring data from different database systems.

A federated architecture is assumed, under which several autonomous local database management systems (LDBMSs) are united for the purpose of data sharing [4, 19]. The LDBMs may have their own local users and applications. It is also assumed that each LDBMs decides which part(s) of its informational content can be contributed to the global system, and presents an export schema to the external users. Although it is assumed that the export schema is relational, other data

models can be accommodated by the inclusion of the necessary data-model translations. The problem of data model translation has been discussed in [20, 34], and practical solutions are available in the form of relational query interfaces to major commercial non-relational DBMSs (e.g., IDMS/R and DATACOM/DB).

Under this approach, the MDBS can be implemented as a front-end, and requires no changes to the LDBMSs. This front-end includes schema information exported from the databases in the MDBS, and a global query interface. The export schemas include information about domains of relation attributes, which can be used to resolve simple domain-incompatibility (e.g. attributes of different types or precision must be compared). In addition, the query language provides operations that tolerate and help resolve relation-definition incompatibility (e.g. incompatible relations corresponding to the same entity type must be merged). Since different sites may require different relational query languages, the front-end must also provide translators between the global and the local languages. This approach allows site queries to be synthesized directly in a language of a member database, and also permits the decomposition of a global query into site queries. The approach is based on an extended relational model which will be introduced below.

5.2 Global Processing

Various methods for processing global queries in heterogeneous distributed database systems have been discussed in, among others, [8, 20, 25, 33, 35]. In general, a global query in MDBS can be processed by specifying a collection of local subqueries and a global query evaluation plan. A local subquery specifies the data to be obtained from local queries. Before a local subquery is sent to its target site for evaluation, it may need to be translated into the target database query language.

Although a federated environment is assumed in this discussion of query evaluation, this assumption does not imply that the software to process global transactions must be distributed throughout the system. Sites may have different functional scopes. In particular, there can be assumed the existence of a specialized site (or sites) that can carry out decomposition and translation of global queries on behalf of other sites. Other sites may originate a global query and forward it to a site that is capable of processing global queries. This fact, of course, may enable the "small" sites (possibly microcomputer-based) to join an MDBS and not only make its data available to other sites but also originate global transactions. A site may also passively participate in an MDBS in simply making its data available to global users.

The first phase of processing a global query involves its decomposition into site subqueries. In the case of a homogeneous relational environment, the goal of query decomposition is to convert the global query into a coherent collection of elementary relational operations on underlying local relations or fragments. Query decomposition has been addressed, in this context, for example in [4, 39]. In particular, [4] discusses equivalence transformations between expressions of extended relational algebra, correctness of distributed query processing strategies, and heuristics that lead to efficient evaluation of distributed queries. Most query decomposition algorithms proposed in the literature for homogeneous DDBMS decompose a query into a sequence of basic relational operations, such as semi-joins, and data moves. In the further discussion, it is assumed that a member database system may support remote query access and that its local DBMS does not need to be modified in order to contribute its data to the federated database system.

The applicability of different query decomposition strategies depends on the level of schema integration. A traditional approach to query decomposition and optimization, based on estimation of volume of data transfer and communication costs, is suitable for systems in which queries are formulated on the assumption of a global integrated schema. Depending on the

scope of functions provided by the site DBMSs, different plans for evaluating a compound query may become feasible. Query-decomposition algorithms based on this approach, which are suitable for heterogeneous databases are, described in [31]. The implementation of this approach requires a separate decomposition module, which is activated after the query is formulated, to derive the "optimal" evaluation plan.

An alternative and novel approach, which is particularly suitable when the query is formulated by directly manipulating the export schemas in the manner described above, can be implemented. In this approach, instead of decomposing a global query, the system synthesizes a global query. Synthesis facilitates incremental construction of individual site queries and the global query evaluation plan. With this method every step in query formulation results in a complete and valid query. Thus the incremental compilation is facilitated, since each change made to a query diagram corresponds to a modification of a complete and valid query evaluation program.

Once a set of site subqueries corresponding to the global query is obtained, the corresponding query evaluation plan can be derived and executed. For example, in the OMNIBASE [32] prototype the plan is derived as a program in a special

Distributed Operation Language, and is executed by a query evaluation supervisor. The supervisor communicates with the server processes at member databases, and instructs them to execute local operations and to transfer results in pipeline or batch modes. Depending on the load of the system, characteristics of the various component database systems and their criteria, stated optimally, different evaluation programs can be generated for a given query.

5.3 Local Processing

Any subqueries that are brought to the participating databases must be stated in the same language as that used by the particular database. The translators must translate the queries from the global format used by the global dictionary directory to the local description where the query will ultimately be processed. The auxiliary databases are used by the translators to perform the mapping from the global to the local formats: the auxiliary databases should therefore be either located or distributed to each of the local database sites. Any results that are retrieved from the participating databases then need to be retranslated to the global format before they are forwarded to the global query site.

The global query site is essentially the hinge interaction between the global data and local databases that contain the data.

In the strategy defined in the global optimization process, the global query site is responsible for the management order in which the participating database are queried. Some overlap may exist if the query could be composed at the global level, or passed as a query to a local database, but the intermediate results are returned to the global query site without concern for details of the local databases. No details need to be known regarding the accomplishment of processing, since the local databases themselves are isolated in the structure. This situation tends to preserve and keep site autonomy, which is one of the fundamental criteria in a heterogeneous environment. A number of systems, such as Multibase [20], impose local query optimization; the optimal solution from the global viewpoint is attained, although, there are some sacrifice of site autonomy.

Conclusions

This article surveys the up-to-date work done in the distributed heterogeneous database system area, emphasizing some critical research areas. The conviction of the need for a global conceptual schema is central to any work in the area, as seen in the question raised by the federated architecture as to the appropriateness of defining a global schema.

Translators are the main difference between the heterogeneous environment and the homogeneous distributed one. A significant step forward would be to discover the characteristics of a "good" translator that would minimize overhead.

Transaction management offers numerous research opportunities, although some problems need to be addressed since little work has been done in this area. It is important to determine what amount of autonomy would be sacrificed through permitting transaction management on multidatabase systems. A technique must be developed to provide correct transaction management in such a system, a step which may require new formalism to assist in describing correctly performed conditions.

The problem of graphical schema integration and query formulation in a federated database system has been discussed. Under the system architecture assumed, an MDBS can be implemented as a front-end to a collection of existing local DBMSs, which need not be altered. This approach is particularly attractive when location transparency is not necessary, or when the MDBS is not stable, because export schemas are frequently added, deleted, or changed. The approach supports and encourages incremental growth.

The export schemas of individual databases are assumed to be globally available. The schemas are represented by diagrams and can be manipulated both by database administrators and end users to integrate schemas, construct views, or formulate queries.

An extension to the relational data model presented, by defining connectors, was also introduced; it can be used to carry additional semantic information about a relational schema. This approach enables definition of new relation-merging operators that tolerate and deal with incompatibility of relations and thus addresses the relation-definition-incompatibility problem.

Query decomposition and translation can be treated within the framework provided by the model. A query-synthesis approach

to the processing of compound queries is another possibility. As relation descriptors are manipulated, site subqueries can be generated and processed through the use of incremental compilation. The site subqueries can be maintained and manipulated directly in the languages of their corresponding target sites.

Finally, a graphical query language that provides unobtrusive assistance to a user formulating a query was discussed. During the process of query formulation, a user is provided with a convenient frame of reference, in the form of a diagram reflecting the current phase of query formulation. Naming and other conventions assist a user in identifying similar entities and attributes, and the formal algebraic foundation makes it possible to keep the user informed about possible errors. None of this precludes an efficient implementation of the language since queries can be optimized and processed by a standard relational interfaces. Since different levels of distribution transparency are allowed it is possible to take full advantage of the user's understanding of a database.

Research opportunities in the heterogenous system areas are plentiful. The solutions to existing problems will lead to a fuller integration of different databases throughout an organization.

References

1. P.A. Bernstein and N. Goodman. "Concurrency control in distributed database systems," ACM Computer Surveys, 13(2), 1981.

2. Y. Breitbart, and A. Silberschatz. "Multidatabase update issues," Proceedings of ACM SIGMOD Conference, Chicago, June 1988.

3. A.F. Cardenas. "Heterogeneous distributed database management: the hd-dbms," Proceedings of the IEEE, 75(5): 588-600, May 1987.

4. S. Ceri and G. Pelagatti. "Distributed Databases - Principles and Systems, McGraw-Hill, 1984.

5. E. Codd. "Relational model of data for large shared data banks," Communications of the ACM, 13(6), June 1970.

6. C.J. Date. Introduction to Database Systems. Addison Wesley, 4 edition, 1982.

7. U. Dayal, "Processing queries over generalization hierarchies in a multidatabase system," Proceedings 9th VLDB, Florence, Italy, 1983.

8. C. Devor and J. Weeldreyer, "DDTS: a testbed for distributed database research," Proceedings of the SCM Pacific 80 Conference, pp. 86-94, San Francisco, California, November 1980.

9. C. Devor, R. Elmasri, and S. Rahimi, "Design of DDTS: a reliable distributed database testbed systems," Proceedings of the 2nd IEEE Symposium on Reliability in Distributed Systems, Pittsburgh, Pennsylvania, July 1982.

10. W. Du and A. Elmagarmid, "Quasi Serializability: A Correctness Criterion for Global Concurrency Control in Interbase," In Proceedings of the International Conference on Very Large DataBases, Amsterdam, The Netherlands, August 1989.

11. W. Du, A. Elmagarmid, and W. Kim, "Maintaining Quasi Serializability in HDDBSs," Technical Report SERC-TR-75-P, Software Engineering Research Center, Purdue University, 1990.

12. W. Du, A. Elmagarmid, and W. Kim, "Affects of Local Autonomy on Heterogeneous Distributed database Systems," Technical Report ACT-OODS-EI-059-90, MCC, 1990.

13. W. Du and A. Elmagarmid, "Maintaining Transaction Consistency in HDDBSs," Technical Report CSD-TR-696, Computer Sciences Department, Purdue University, 1990.

14. W. Du and A. Elmagarmid, "Preserving Data Integrity in HDDBSs," Technical Report CSD-TR-970, Computer Sciences Department, Purdue University, 1990.

15. A. K. Elmagarmid and Y. Leu. "An optimistic concurrency control algorithm for heterogeneous distributed database systems," A quarterly bulletin of the Computer Society of the IEEE technical committee on Data Engineering, 10(3):26-32, September 1987.

16. A. K. Elmagarmid and W. Du. "A Paradigm for Concurrency Control in Heterogeneous Distributed Database Systems," In Proceedings of the Sixth International Conference on Data Engineering, Los Angeles, California, February 1990.

17. A. K. Elmagarmid and W. Du. "Supporting Value Dependency for Nested Transactions in Interbase," Technical Report CSD-TR-885, Purdue University, May 1989.

18. A. Ferrier and C. Strangret, "Heterogeneity in the distributed database management systems Sirius-Delta," Proceedings of the 8th VLDB International Conference, Mexico City, Mexico, September 1982.

19. D. Hiembigner and D. McLeod. "A federated architecture for information management," ACM Transactions on Office Information Systems, 2(2), July 1985.

20. T. Landers and R.L. Rosenberg. An overview of multibase. In H.J. Schneider, editor, Distributed Databases, North-Holland, 1982.

21. J.A. Larson. "Bridging the gap between network and relational database management systems," Computer, 16(9):82-92, September 1983.

22. Y. Leu, A. Elmagarmid, and M. Rusinkiewicz, "An Extend Transaction Model for Multidatabase Systems," Technical Report CSD-TR-925, Department of Computer Science, Purdue University, 1989.

23. Y. Lien. "Hierarchical schema for relational databases," ACM Transactions on Database Systems, 6, 1981.

24. Y. Lien. "On Equivalence of Database Models," Technical Report Database Res. Rep. 3, Bell Labs, Homdel, NJ., 1980.

25. W. Litwin. "MALPHA: a relational multidatabase manipulation language," EUTECO Proceedings, North-Holland, 1983.

26. W. Litwin and A. Abdellatif, "An Overview of the Multi-Database Manipulation Language MDSL," Proceedings of the IEEE, May, 1986.

27. S. Navathe, T. Sashidhar, and R. Elmasri, "Relationship merging in schema integration," Proceedings of the Tenth International Conference on Very Large Data Bases, Singapore, August 1984.

28. M.T. Ozsu and P. Valduriez. Principles of Distribute Database Systems. Prentice-Hall, 1991.

29. R. Rajinikanth, M. Jacobson, G., Lafond, C., Papp, W., and Piatetsky-Shapiro, G., "Multiple database integration," Proceedings of the first International Conference on System Integration, April 1990.

30. M. Rusinkiewicz and B. Czejdo, "Query transformation in heterogeneous distributed database systems," Proceedings of 7th International Conference on Distributed Computing, Denver, Colorado, 1985.

31. M. Rusinkiewicz and B. Czejdo, "An approach to query processing in federated database systems," Proceedings of the Twentieth International Conference on System Sciences, Kailua-Kona, Hawaii, January 1987.

32. M. Rusinkiewicz et al. "Query Processing in OMNIBASE - A Loosely Coupled Multi-Database System. Technical Report UH-CS-88-05, University of Houston, Department of Computer Science, February 1988.

33. J.M. Smith, P.A. Bernstein, U. Dayal, N. Goodman, T. Landers, K. Lin, and E. Wong. "Multibase-integrating heterogeneous distributed database systems," In Proceedings of the National Computer Conference, pages 487-499, AFIPS Press, Reston, Va., May 1981.

34. S. Spaccapietra, B. Demo, A. Di-Leva, C. Parent, C. Perez de Celis, and K. Belfar. "An approach to effective heterogeneous database cooperation," In R.P. van de Riet and W. Litwin, editors, In Distributed Data Sharing Systems, North-Holland, 1982.

35. M. Templeton, D. Brill, E. Lund, P. Ward, A.L.P. Chen, and R. MacGregor, "Mermaid - A Front-End to Distributed Heterogeneous Database," Proceedings of the IEEE, pp. 695-704, May 1987.

36. D. Tsichritzis and A. Klug. "The ANSI/XS/SPARC DBMS Framework Report of the Study Group on Database Management Systems," Pergamon Press, Toronto, Canada, 1978.

37. D.C. Tsichritzis and F.H. Lochovsky. "Data Models. Prentice-Hall, 1982.

38. J.D. Ullman. "Principles of Database Systems. Volume 2, Computer Science Press, Potomac, Maryland, 1982.

39. Y. Vassilou and F. Lchovsky. "Dbms transaction translation," In Proceedings COMPSAC 80, IEEE Computer Software and Application Conference, 1980.