"In nature there is fundamental unity running through all the diversity we see about us".

Mahatma Gandhi (1924)

"With the increasing use of databases, we expect the integration problem to be more severe and pervasive. New technologies of networking, distributed databases, knowledge-based systems, and office systems will tend to spread the shared use of data in terms of number of users, diversity of applications, and sophistication of concepts. Design, manufacturing, and engineering applications are becoming centered around database management systems. The need for methodologies for integrating data in its diverse conceptual and physical forms is thus expected to increase substantially".

C. Batini, M. Lenzerini, and S.B. Navathe
A comparative Analysis of Methodologies for Database Schema Integration [BAT86]

A paper based on this chapter, entitled "A DHDBMS Architecture" was presented at the International Conference on Systems Management'90 at Hong Kong in June 1990 and the same has been published in the proceedings of this conference entitled "The Impact of Information Technology on Systems Management" [GOY90a].

40
2.1 Introduction

The database approach to data processing requires that all the data relevant to an enterprise be stored in an integrated database. By 'integrated' we mean that single conceptual schema describes the entire database, that all accesses to the database are expressed relative to that schema, and that such accesses are processed against a single (logical) copy of the database. Unfortunately, in the real world many databases are not integrated. We have discussed in Chapter 1 that often, the data relevant to an enterprise is implemented by many independent databases, each with its own conceptual schema. Such databases are nonintegrated. We have also discussed that these databases may be managed by different DBMSs, perhaps on different hardware. Therefore, in addition to being nonintegrated the databases are distributed and heterogeneous.

A principal problem in using databases of this type is that of integrated retrieval. In such databases, each independent database has its own schema, expressed in its own data model, and can be manipulated only by its own data manipulation language. Therefore, the most ambitious requirement of a DHDBMS is the capability of providing a superview [DAY84], [MOT87] of the system which is transparent not only to data distribution, but also to heterogeneity of DBMSs. Superview is a single global
conceptual schema which is formed by integrating the conceptual schemas of independent databases (or portions thereof) that will accommodate the new application. Such type of a system suppresses differences of DBMSs, languages, and data models among the databases and provides users with a common data model and a single query language. This chapter discusses a DHDBMS architecture based on the concept of superview.

2.2 DHDBMS Design Objectives

A DHDBMS should have three key design objectives: generality, compatibility and extensibility [SMI81]. To satisfy the first objective, a DHDBMS should be designed as a general tool, capable of providing integrated access to various database systems used for various applications. The geographical distribution of the database should be transparent to the user.

The fundamental need for uniform, integrated access to such databases arises because users cannot be expected to learn the use of many different DBMSs and the operational differences between them. To achieve effective interconnection of remote, heterogeneous DBMSs, the users must have uniform, integrated access to the different DBMSs. This feature is necessary because the component DBMSs are existing systems and these were not originally designed to be part of a DHDBMS.
The second requirement is that a DHDBMS should co-exist and be compatible with existing database systems and applications. It should be understood that the component DBMSs are already in operation, and, therefore, these cannot be easily modified. As the local sites retain full autonomy for maintaining the databases, all local accesses and application programs should continue to operate without any change under the DHDBMS.

The third design objective is that it must be relatively easy to add and remove systems to and from the DHDBMS as the component DBMSs are not fixed permanently at the design time.

2.3 DHDBMS Functions and Requirements

With the above design objectives, it is clear that the preexisting component DBMSs would be integrated in their present working form through a DHDBMS. A DHDBMS is designed to provide a logically integrated user interface to a physically nonintegrated environment. It provides a single interface through a single global query language to data in all the databases. Through this interface, a DHDBMS presents a user with the illusion of a single, integrated, nondistributed database. It assumes complete responsibility for knowing the location of the local databases, accessing the data at each of the local DBMSs, resolving data incompatibilities, and combining the data to produce a
To correctly and efficiently execute multi-database queries, a DHDBMS has to perform the following functions [LAN82].

1) Accept a query expressed in the user's global query language;
2) Translate this query into a set of subqueries expressed in the different languages supported by the local DBMSs;
3) Formulate an efficient plan for executing a sequence of subqueries and data movement steps;
4) Implement a plan for accessing data at individual local sites;
5) Move the results of subqueries at a site for integration;
6) Resolve incompatibilities between the databases, such as differences in data types and conflicting schema names;
7) Resolve inconsistencies in copies of the same information that are stored in different databases; and
8) Combine the data into a single answer.

Before local databases can be accessed through a DHDBMS, the local host systems must be connected to a communication network. This network can be local or geographically distributed. It (DHDBMS) is then connected
to the same communication network, as illustrated in Figure 2.1. After this, a global user can access data in the local databases through this DHDBMS, using a single global query language. Local application programs can continue to operate using the existing local interfaces, as before.

There are many important issues in designing a DHDBMS such as Query Language, Query Processing Algorithm, Data Structure, Data Model, Concurrency Control, Directory Management, Recovery Mechanism, Integrity and Security. Each topic listed above is an important research area itself in a centralized DBMS environment. In a DHDBMS, the above issues become more complicated and harder to handle. For example, in the DHDBMS environment, the solutions to deadlock, concurrency control and recovery are much more complex than in the centralized environment. This is because the data, control and status information about the processes in the system are distributed. In addition to these, data translation, data replication and fragmentation, operating system coordination, location and replication transparency, reliable communication are all problems specifically related to a distributed database environment. Our purpose is to discuss the access control problem in a DHDBMS.

The above issues can be tackled by adopting a multiple schema architecture [DEV82], that provides
Figure 2.1: Communication Network connects a DHDBMS and its global users with local systems.
heterogeneity transparency along with location transparency (Figure 1.5). The use of multiple schemas and mappings between them serves as the mechanism for providing transparency across dissimilar systems and architectures [GLI84], [SPA80].

Several semantic database models are discussed in [HUL87]. Some of these database models are very powerful in data structuring capabilities and can be used to define schemas and mappings between them for a DHDBMS. Four prominent semantic data models are discussed in the next section. Based on these models, a generalized DHDBMS architecture is presented in section 2.5.

2.4 Semantic Database Models

Database management systems have been developed in the past two decades using various data models and architectures. The primary data models used for implementation are the hierarchical, network and relational models. More recently, several semantic data models, significantly more powerful than primary models have been proposed [HUL87].

We have discussed in chapter 1 that semantic data models were first introduced primarily as schema design tools because they embody the fundamental kinds of relationships arising in typical database applications. The schema so designed was then translated into one of the
traditional models for ultimate implementation. The field of semantic models is continuing to evolve. Consequently, semantic models are more complex than common data models. Therefore, there has been increasing interest in using these models as the full-fledged database management systems and as computer front-ends to existing systems. In this thesis, we are interested to use these models as a common front-end to heterogeneous DBMSs, so that a user can be provided integrated access facility to these DBMSs through this front-end.

If a semantic database model supports all the features available in relational, hierarchical and network data models of these DBMSs, then a global conceptual schema can be formed by integrating the conceptual schemas of these independent databases. It is highly probable that a semantic database model may not support all the features of underlying data models (corresponding to component DBMSs) and therefore some transformation may be required to convert local conceptual schemas into intermediate schemas and then these intermediate schemas are integrated to form a global conceptual schema. Some sort of mapping will have to be provided between global conceptual schema and local conceptual schemas. In this section we are analysing the capabilities of four prominent general purpose semantic database models for the purpose of schema integration. Schema integration is defined as the activity of integrating the schemas of existing or proposed databases into a global
conceputal schema. This global conceptual schema is a virtual view (called superview) of all databases taken together in a distributed database environment. Different methodologies for database schema integration are discussed in [BAT86].

We would also discuss whether these models provide facilities for defining views because we would use view mechanism for access control at the global level. The semantic models discussed here are (1) The Entity-Relationship Model, (2) The Functional Data Model, (3) The Semantic Database Model, and (4) The Semantic Association Model.

2.4.1 The Entity-Relationship Model

[CHE76] discusses strengths and weaknesses of the network and the relational models. It states that the network model provides a more natural view of data by separating entities and relationship (to a certain extent), but its capability to achieve data independence has been challenged. It also states that the relational model is based on relational theory and can achieve a high degree of data independence, but it may lose some important semantic information about the real world.

[CHE76] presents a data model, called the Entity-Relationship (ER) model and claims that this model
has most of the advantages of the relational and the network models. The ER model, proposed by Chen in 1976, is considered to be one of the first semantic database models to appear in the literature, although the term "semantic" was not in use at that time. Schemas of this model have a natural graph-based representation and support the representation of abstract sets of entities, relationships between these entity sets, and attributes defined from both entity and relationship sets to printable values. An entity is a class of objects of the real world having similar characteristics and properties. A relationship is a class of elementary facts or associations among entities. An attribute is an elementary property either of an entity or a relationship. Thus, an ER schema consists of types and relationships interconnecting these types, along with printable attributes of the types and relationships. Relationships can be restricted to 1:1, many:1, or many:many, and attributes can be restricted to 1:1.

The ER model can be used as a basis for a unified view of data. [CHE76] uses the ER model as a framework and illustrates by an example, that from this model, the relational and network data models may be derived. In essence, the ER model may be viewed as a generalization or extension of the relational and network models. Therefore the ER model can be used for schema integration. In fact, the development of a prototype heterogeneous front-end system called Integrated Information Support Systems (IISS),
Figure 2.2: DHDBMS Architecture based on the ER Model
sponsored by the US Air Force is based on a model similar to the ER model. This system has a goal of accessing databases on three types of hardware (VAX, IBM, and Honeywell) and many DBMSs. It currently accesses data in Oracle on the VAX and IMS on the IBM. Their approach is batch oriented. There is no adhoc query language and all retrieval requests must be precompiled [TEM86].

[CHE76] proposes use of SQL-like language [DAT83] for data retrieval, insertion, deletion and updation. By using SQL-like language, views can be defined by the technique discussed in [CHA75]. [WAN87] has discussed an access control mechanism in a DHDBMS. The DHDBMS architecture used in [WAN87] is based on the ER Model and its global conceptual schema is a set of entities and relationships among the entities. It uses a query language with NQUEL syntax [DAY82] to illustrate examples. NQUEL is based on the relational query language QUEL [DAT87]. Both NQUEL and QUEL have features to define views. A DHDBMS architecture based on the ER model is shown in Figure 2.2.

2.4.2 The Functional Data Model

The Functional Data Model (FDM) was introduced in 1976 by Karschberg and Pacneco [HUL87] and is recognised as the first semantic model centered around functional relationships, that is, attributes. Like the ER model, a considerable amount of research has developed around FDM,
and several other semantic models have adopted the attribute-based approach.

The basic constructs of the Functional Data model are entities and functions; these correspond to conceptual objects and their properties. Entities with similar properties are grouped together to form entity sets. Entity functions can be single-valued or multi-valued. When applied to a given entity, a function returns a specific property associated with that entity. Each property is represented by either a single value or a set of values [SHI81]. FDM also supports ISA relationships [HUL87]. An ISA relationship from a type SUB to a type SUPER indicates that each object associated with SUB is associated with the type SUPER.

The data language DAPLEX [SHI81] for this model was the first integrated data definition and access language formulated entirely in the high-level terms. DAPLEX was also the first database access language to give a prominent role to attributes, permitting their direct usage and also the use of their inverses and compositions. Like the other languages in this class, the query specification portions of this language contain syntactically elegant renditions of most of the basic elements of the first order predicate calculus and is thus fundamentally nonprocedural [HUL87].
In FDM, global conceptual schema is composed of entity types and functions between entity types. Each entity type contains a set of entities, so functions map entities into entities. The FDM embodies the main structures of both the flat file data models, such as the relational model, and the link structured data models, such as CODASYL. Entity types correspond roughly to relations in the relational model or record types in the CODASYL model. Therefore, the FDM can be used as a front-end distributor to heterogeneous DBMSs. In fact, the FDM has been used to provide a unified interface to distributed heterogeneous databases in the MULTIBASE project [LAN82], [SMI81] sponsored by Computer Corporation of America. Integration of schemas is studied in [DAY84]. Through an example, [LAN82] illustrates use of MULTIBASE for integrating four heterogeneous DBMSs. Two of these DBMSs are built using a file model and the remaining two are based on the network and the relational model.

DAPLEX also supports Derived Data Definitions as an integral part of a semantic model. Derived Data Definitions are one of the fundamental mechanisms in semantic models for data abstraction and encapsulation. Derived data are closely related to the notion of a user view (or external schema), except that derived data are incorporated directly into the original schema rather than used to form a separate new schema. Another difference is that a view may contain raw or underived components, as well
Figure 2.3: Multibase Architecture
as derived information. As it supports view mechanism, an access control mechanism can be implemented in a DHDBMS which is based on the FDM architecture. The schema integration architecture of the MULTIBASE is shown in Figure 2.3.

2.4.3 The Sematic Database Model

The Sematic Database Model (SDM) [HAM81] was among the first published models to emphasize the use of the grouping constructs and the support of derived schema components. In particular, derived schema components permit data relativism, that is, multiple perspectives on the same underlying data set. SDM is unique in that it provided a rich set of primitives for specifying derived attributes and subtypes. For example, subtype relationships in SDM are broken into four categories: (i) those that are attribute, (ii) those defined by set operations (e.g. intersection) on existing types, (iii) those that serve as the range of some attribute, and (iv) those that are user specified.

The richness of SDM as a schema specification language highlights the trade-off in sementic modelling between providing a small or large number primitive data structuring constructs. In models with a small number of constructs, the representation of some data sets requires the artificial combination of these constructs; in a model
with many constructs such as SDM, the designer is continually forced to choose from among a variety of ways of representing the same data. SDM, like FDM and unlike the ER model, is centered around attributes, but is richer (and the more complex) than either FDM or the ER mode. There is a data manipulation language associated with the SDM. It supports view definition facility.

As discussed above, SDM is a powerful data model which consists complex data structuring capabilities. It also has the common data structure features available in primary data models. Therefore, SDM can also be used as a front-end distributor to heterogeneous DBMSs. As the data manipulation language associated with this model supports view mechanism, an access control mechanism can be implemented in a DHDBMS which is based on SDM. We have not come across any DHDBMS prototype (which is being developed) based on this model during our literature survey.

2.4.4. The Sementic Association Model

The Sementic Association Model (SAM*) [SU83] has been developed to support full-fledged database applications. SAM* is rich in data sements and contains a number of constructs for modelling the relationship among the data found in engineering, business, scientific and statistical databases.
The model distinguishes different users of some of the fundamental structural constructs of semantic models and in some cases provides them with different update semantics. As a result, the model supports a limited form of data relativism whereby a given construct might be viewed as having two or more different underlying structures within the same schema.

The basis for SAM* schemas is provided by atomic concepts [SU83]. These include integer, real, character-string, and Boolean types, as well as structured programming language data types constructed from these, including vectors, arrays and ordered and unordered sets, time series and text.

In SAM* physical or abstract objects are represented using nonatomic concepts. Seven kinds of associations or type constructs for building these concepts are used. The different associations have somewhat specific intended uses, and operations especially tailored to them. The seven kinds of associations are (i) membership association, (ii) aggregation association (iii) interaction association (iv) generalization association, (v) cross-product association (vi) composition association, and (vii) summarization association.

SAM* has been selected as the data model for the Integrated Manufacturing Database Administration System.
The IMDAS supports an integrated database composed of physically distributed databases. The SAM* model contains several semantic constructs to support the description of data partitions, constraints and mappings needed for distributed functionality.

To describe the mappings between the logically single integrated database and the numerous physically distributed databases, three levels of view definition are by the IMDAS. These views are: (1) global external view, (2) global conceptual view, and (3) fragmented view. A global external view defines a segment of the integrated database as seen by a single control process. It identifies the data entities and associations used by the control process. There can be many global external views defined for the various control processes. The global conceptual view (superview) is the integration of all the external views and represents a comprised view of the factory database. Fragmented views represent global conceptual data which are physically partitioned or replicated across the component systems. They describe the distribution of the conceptual database; data within each component system is represented as a distinct fragmented view. Each fragmented view maps into the data representation supported by one local data management facility on one component system.
**Figure 2.4: IMDAS Architecture**
The IMDAS Global Data Manipulation Language (GDML) [KRI85] has been designed to support the SAM* data model [BAR86] and is strongly based on the draft ANSI standard database manipulation language. Although details of the IMDAS GDML is not available as we could not get [KRI85], it is hoped that this data manipulation language provides facility to define external views dynamically. Therefore, an access control mechanism can be implemented in a DHDBMS based on SAM*. The IMDAS architecture is shown in Figure 2.4.

2.4.5 Some Other Models

We have discussed earlier that transparent access to a variety of DBMSs requires the definition of a data model and data manipulation language which can be used to mediate between different DBMSs. The language and model need to be broad enough to cover all potential models used in component DBMSs and all functionality of the component languages, or at least a subset of the component language which will be supported in the distributed system.

Several DHDBMS prototypes have been developed using the relational model technology to define global conceptual schema. Some of these prototypes are: Multics Relational Data Store Multibase (MRSDM), [WON85], Amco Distributed Database System (ADDS) [BRE85], AIDA [TEM86],
MERMAID [TEM87a], SIRIUS-DELTA [FER82], [ESC86], Network Data Management System (NDMS) [STA85], [DEE87b]. Some of these prototypes use extended ANSI/SPARC architecture discussed earlier and support SQL type data manipulation language. Most of these prototypes support view mechanism and therefore access control mechanism using view mechanism can be implemented in these systems.

2.5 General DHDBMS Architecture

We have emphasized earlier that the major technical challenge in supporting distributed heterogeneous database is to provide the user with an integrated, coherent, global view (superview) of the data stored in the local databases. We have also explained that a user of the global database must be insulated not only from the locations of the data, but also from local system details like data model, data description, and query language. We have also seen that there are several architectures which may meet requirements.

In the preceding section, we have discussed some DHDBMS architectures based on different data models. Although there are some variations in these architectures, but on the whole all of them use similar approach of using a data model as a front-end to component DBMSs. Similar type of DDBMS architectures are also discussed in [DRA80], [SPA80], [SPA82a], [DEV82]. In this section, a generalized
DHDBMS architecture is presented. We have not chosen a particular data model for this architecture. We will use this architecture to discuss access control mechanism. Though individual prototypes which we have discussed earlier, may not match this architectural framework exactly, but the architecture presented here supports all the necessary features of a DHDBMS. The five schema architecture presented here is basically an extension of ANSI/SPARC three-schema architecture used for centralized database management systems. The five schemas are:

(i) Global External Schemas (Global User Views);
(ii) Global Conceptual Schema (Superview);
(iii) Local External Schemas (Local User Views);
(iv) Local Conceptual Schema; and
(v) Internal storage schema.

The architecture is shown in Figure 2.5. The arrows in the figure denote mappings between different schemas. In the following sections, we describe the contents and characteristics of the above schemas.

2.5.1 Global External Schema

A global schema is a global user view schema. This schema describes the portion of the database that a particular user is permitted to process. This schema contains the application oriented description of the database and database entries for a global database user.
Figure 2.5: General DHDBMS Architecture
There can be many global external schemas corresponding to many users.

2.5.2 Global Conceptual Schema (Superview)

The global conceptual schema (Superview) is created based on a global data model. The global conceptual schema needs to capture the meaning of total data in the environment. It contains the description of the structure and relationship of the data within the global database. It contains the semantic and syntactic constraints for the database. It specifies the integrity requirements in terms of data and transactions for the global network. The specification of authorization and security enforcement also resides in this schema. The data of a DHDBMS may be centralized, partitioned, replicated, or a hybrid of these. The description of data distribution within the database is retained in this schema. The schema also describes query language translation procedures between the global and local data models. It also contains an auxiliary database, which is used for resolving various conflicts [DAY84], and for integrating partial results received from different sites for a multi-site query.

A DHDBMS like MULTIBASE may support more than one global conceptual schema for the same underlying DBMSs. In this case a particular global schema may be convenient for a particular group of global users and so on. In this thesis,
access control mechanism is discussed based on a single global conceptual schema, although it can be similarly applied when more than one global conceptual schemas are present in a DHDBMS environment.

2.5.3 Local External Schema

This is equivalent to the external schema of the ANSI/SPARC three-schema architecture for a single site of the network. It contains the user entries and applications for the database at the local site. Local users have access to local DBMS through this schema.

2.5.4 Local Conceptual Schema

The data characteristics and constraints in this schema are similar to those of the global conceptual schema except that this schema refers only to data within the local database.

2.5.5 Internal Storage Schema

This schema contains the description of the internal format of the data in a local database and its mapping to physical storage. This schema will be site dependent.

2.6 Functional Specification of a DHDBMS

The functional structure of a DHDBMS conforms to
the architecture described above. In each site, the functional system is divided into four functional components: (1) Global Functional System, (2) Local Functional System (3) Communication Network System and (4) Local Database Interface. Each site has the identical Global Functional System as far as functionality is concerned. The Local Functional System at different sites may be different. The Communication Network System and Local Database Interface serve as an interface between Global Functional System and a Local Functional System. Figure 2.6 illustrates the relationships between these four functional components. Each is discussed below.

2.6.1 Global Functional System

The Global Functional System provides a user interface. All the queries of global users formed over global external schemas go through this interface. The system parses the global query. It provides a mechanism for checking of user authorization. Once this is satisfactorily done, it performs the task of global external schema to global conceptual schema mapping referred in Figure 2.5. This operation modifies the global query into an intermediate query based on the global conceptual schema. This intermediate query is broken into a set of subqueries to be processed at the various sites of the system. It also evaluates the processing cost of a query and selects a processing plan which will minimize execution cost based on
Figure 2.6: Functional Components of each site in a DHDBMS
some set of cost formulas. After evaluating the processing cost, it generates a global execution plan for the subqueries and distributes the subqueries to the designated sites with the help of the Communication Network System. It also handles global recovery facilities to maintain the reliability of the system. The Global Functional System at the query origination site is responsible for the query and will coordinate processing of the query by integrating the results of subqueries and issuing appropriate responses to users. All these operations are performed by different components of the Global Functional System. These components are referred as Parser, Decomposer, Transaction Handler, Transaction Optimizer, Execution Plan Generator etc. To perform all these operations, the Global Functional System contains a lot of information such as contents of global conceptual schema, global transaction management information, schema to schema object mappings, global catalogue information, data authorization requirements and data integrity requirements.

2.6.2 Communication Network System

The Communication Network System handles data communication and transmission among the sites of the network. Thus, the Communication Network System provides the interface between the Global Functional System and the Local Functional System of each site.
2.6.3 Local Database Interface

The Local Database Interface accepts a request from the Communication Network System and creates a transaction or site query (corresponding to a subquery) for the Local Functional System to process it. It also performs data and query translation whenever necessary. Since we assume that each Local Functional System is a complete DBMS operating independently from the Global Functional System, an interface between both these systems is required. The Local Database Interface component of each site serves as this interface. When a request is sent from the Global Functional System for processing at the Local Functional system, the Local Database Interface accepts the request and sets up a user process executing on behalf of the query owner to initiate the request to the Local Functional System. The Local Functional System performs the required operations to complete the request and returns the result (or messages) back to the Local Database Interface through the user process. The Local Database Interface passes the result to the Communication Network System for transmission to the destination site and kills the user process it created.

2.6.4 Local Functional System

The Local Functional System is essentially a complete centralized DBMS. Unlike the Global Functional System, the Local Functional Systems may be different at
Each Local Functional System accepts a query from the Global Functional System (through the Communication Network System and Local Database Interface) which is to be executed at that site, and returns the result to the Global Functional system. The Local Functional System does not distinguish between queries from the Global Functional System and queries entered directly by users at that site. To the Local Functional System, the Global Functional System is simply a source of queries to be processed.

2.7 Query Language

We have presented a general DHDBMS architecture based on a semantic database model in section 2.5. A query language is also required for data manipulations. There are several languages that have been proposed for different data models [HUL87]. We will use NQUEL for examples [DAY82]. NQUEL is an extension of QUEL [DAT87] and has not been described here. Details of NQUEL can be seen in [DAY82]. We use this language merely as an example. The other advance data manipulation languages such as DAPLEX [SHI81] can also be used for illustrating examples.

2.8 Query Processing in a DHDBMS

In this thesis our main concern is access control issues for a DHDBMS. Query processing affects the access control, hence we will briefly discuss query processing in this section.
In general, there are two steps for processing a query. The first step is to generate a global execution plan and the second is to execute the global execution plan. Depending on the goal of the query processing algorithm (e.g., minimizing execution cost, minimizing data transmission between sites, minimizing response time, or minimizing the number of messages) different approaches to query processing are required.

Many query processing algorithms have been proposed for achieving different processing goals [LAN82],[DAY83],[DEE84],[LOH84]. We do not intend to propose a new query processing algorithm. Instead, our purpose is to describe a query processing framework for a DHDBMS in such a way that many of these algorithms will fit into the framework.

The steps for constructing an execution plan can be described as follows. The query is parsed to produce a parse tree. The parsed query tree contains the internal names of the data in the data base. The query tree is decomposed into a sequence of subtrees which will be processed at different sites. From these subtrees several feasible execution plans may be produced. An optimal global execution plan is selected from these plans.

A global execution plan is in fact a sequence of subplans. Each subplan is sent to a specified site. The
distribution of the subplans is done through the Communication Network System.

Each Subplan contains a subquery which is to be executed by a local system. (This subquery may have to be translated into a site query by Local Database Interface. It would depend upon the type of data model chosen). This subquery is processed as a local query at the local system. The data received from different sites (corresponding to subqueries) are integrated and the final result is passed to the user.

2.9 Summary

In this chapter, we discussed DHDBMS design objectives and need for a DHDBMS architecture and its functional requirements. We also discussed about some semantic database models and DHDBMS prototypes being developed using these models. We briefly described the architectures of these prototypes and presented a generalized DHDBMS architecture. We also talked about query language and query processing mechanism in a DHDBMS. Based on this architecture, query language and query processing mechanism, a mechanism of access control in a DHDBMS is discussed in Chapter 4. In chapter 3, access control mechanisms for existing DBMSs are discussed.