Chapter 2

The Shoulders of Giants

The task of providing medical decision-support applications with access to patient data that are stored in disparate clinical information systems is a necessarily complex undertaking. Clinical data consist of numerous, varied patient parameters that are richly interrelated. Clinical queries relevant to decision support may involve complicated search criteria and may return complex data objects. Implemented query models for clinical data vary along several dimensions. In order to develop a useful standard query model for medical applications, one must accommodate the complexity and heterogeneity inherent in the electronic representation of clinical data. Doing so requires an understanding of previous research results in medical informatics, in database systems, and in artificial intelligence. My objectives in this chapter are (1) to create a framework for my research by characterizing the kinds of applications that can benefit from it and by defining the terms and concepts that I shall use to describe it, and (2) to review previous research from a number of disciplines that addresses the same problems I am addressing from a different perspective or that constitutes a component of the solution that I propose.

Section 2.1 presents a short glossary of relevant terminology that appears in the remainder of the chapter. Section 2.2 briefly reviews several classes of decision-support applications in medicine, focusing on the various ways in which the applications acquire patient-specific data. Section 2.3 describes current research initiatives by the medical informatics community to standardize the representation of clinical data and to increase the portability of decision-support applications that require automated access to clinical data. Section 2.4 describes current research initiatives in the database and artificial intelligence communities that address the issues of standardization and interoperability that are relevant to medical applications, although in a domain-independent way. In Section 2.5, I outline the individual components required for the query model mapping technology that I introduced in Chapter 1, and I review previous research relevant to each of the components.

2.1 Terminology

The following glossary of database terms provides a reference point for further discussion. Although there may not be universal agreement regarding the definitions of these terms, the glossary was derived from several texts [Gardarin and Valduriez 1989, Ullman 1989, Wiederhold 1983] and represents a consensus among several database researchers.

2.1.1 Basic definitions

1. **Database**: A large, persistent, and shared representation of the data objects and of the relationships among data objects that are relevant to the functioning of some enterprise and that can be accessed by a computer.

2. **Database Management System (DBMS)**: A software system that reliably manages a database and provides access to large collections of data objects efficiently. Note that this is a very generic definition: One may describe everything from the Unix operating system to an object-oriented database with a natural language interface as a DBMS.

3. **Database System**: A database and a corresponding DBMS.

4. **Data Model**: A set of constructs with associated composition rules used to specify the structure of and the constraints upon data in a database. A data model provides an *abstract mathematical formalism* for describing data, which allows users to see information not as raw bits, but in more abstract and intuitive terms.

5. **Data Definition Language (DDL)**: A specification language that allows users to describe the data in a database using a specific data model. Data definition languages are used to specify database schemas (see glossary item 6).

6. **Database Schema**: A complete specification of the object types relevant to an enterprise, expressed using a particular DDL. The information captured in a database schema depends on features of the data model in which it is specified. Specifically, a database schema is *not* necessarily a relational database schema, but may be specified with respect to data models that are more or less expressive.

7. **Data Manipulation Language (DML)**: A language interface provided by the DBMS to specify operations on the database, such as queries and updates. DMLs correspond to particular data models and, as such, may vary in expressiveness and level of abstraction.

2.1.2 Levels of database description and data manipulation

Database systems may be described and manipulated at different levels of abstraction. For example, a database system may have a low-level query language that specifies in assembly language the physical locations of requested data, or a high-level query language that specifies in first order logic the abstract properties of requested data. Different levels exist for two reasons:

1. Humans and computers require different degrees of detail in describing and manipulating data.

2. Higher levels of abstraction provide *independence* from lower-level implementation details that may change with time or may vary across database systems.

Four levels of database description typically exist [Batini, et al. 1992, Sheth and Larson 1990]:

1. *External level*: A description of a *subset* of the database corresponding to a program's or a user's view of the enterprise. An external view typically is available to “client” applications as a circumscribed window on the enterprise data. For example, the portion of a clinical database used by the pharmacy staff comprises an external-level schema of the entire clinical database.

2. *Conceptual level:* An *implementation-independent* description of the entire database corresponding to a global view of the modeled enterprise. A database schema specified at the conceptual level is a *conceptual schema.* For example, a data description of the entire patient record specified using an abstract data model would constitute a conceptual schema. Conceptual-level description hide many implementation details that are of no interest to the users of a database system.

3. *Logical level*: An *implementation-dependent* description of the data that can serve as the interface to an operational database system. A database schema specified at the logical level is specific to the data model used by the DBMS at a particular site and is called a *logical schema*. Importantly, a single conceptual schema may be represented by various logical schemas.

4. *Physical level*: A *representation-dependent* description of the database that corresponds to the physical storage structures and access methods used to store and manipulate data in secondary memory. Data descriptions and data manipulations specified at the logical level typically are translated by the DBMS to corresponding representations and operations at the physical level, so that they can be efficiently implemented and executed by the operating system.

Having established these terms, I now can describe previous research in sharing decision-support applications across heterogeneous database systems.

2.2 Accessing Clinical Data

A review of prominent medical expert systems developed over the past 20 years reveals that decision-support applications acquire patient-specific data in one of four ways:

1. *Via direct interaction with the user*. The system prompts the user for all relevant clinical parameters, sometimes offering synonym dictionaries and phrase-completion facilities to assist in the process. This is especially the case for diagnostic systems that require the entry of complex clinical findings. Examples of such "input-driven" systems are MYCIN [Buchanan and Shortliffe 1984]nd QMR [MILLER AND MASARIE

1990]. A notable feature of input-driven systems is that no patient-specific data are stored between consultation sessions.

2. *Via the loading of special-purpose data files*. The system initially captures patient data via user interaction and stores the data in specially formatted files that are managed by the operating system. During subsequent consultation sessions, the patient-specific files are loaded into the memory space of the application and inference proceeds. An example of such a "file-driven" system is ONCOCIN [Shortliffe 1981]. A notable feature of file-driven systems is that the stored patient data cannot be shared easily among other applications and that all queries and updates are performed by the decision-support system.

3. *Via access to an online clinical database*. The decision-support system queries a general-purpose database system that captures, stores, and retrieves clinical data. Typically, the clinical database supports a number of enterprise-wide functions, including direct patient care, administration, and clinical research. Queries to the database are executed via an application programmer interface (API) that is specific to the clinical database system and that is available to all client applications. Examples of such "database-driven" systems are clinical reminder and alert applications integrated with the following clinical information systems: HELP [Kuperman, et al. 1991], COSTAR [Barnett, et al. 1982], TMR [Stead and Hammond 1988], and the Regenstrief Medical Record System [McDonald 1984]. A notable feature of database-driven systems is that the APIs used to access patient data are complex and very specific to the local clinical database implementation [Adams 1986]; as a result, decision-support applications cannot be shared across multiple databases because each database's API is significantly different. These applications suffer from the portability problem discussed in Chapter 1.

4. *Via access to abstracted data sets*. The system queries a longitudinal database that stores clinical data abstracted from the medical record systems of numerous provider sites. The data usually are related to a single clinical syndrome, such as rheumatism or coronary artery disease. Data can be pooled from multiple sources because a common

> set of clinical parameters and a common descriptive vocabulary are specified *a priori* and are enforced during the data-abstraction process. An example of such a "databank-driven" system is ARAMIS (The American Rheumatism Association Medical Information System) [Blum 1981]. A notable feature of databank-driven systems is that, although they can access data that are primarily stored in a variety of medical record systems (including clinical databases), the data must first be abstracted into a uniform and standard representation, a generally laborious process.

A standard query model of clinical data and an automated query translation mechanism for clinical queries would benefit decision-support systems in each of the four categories by providing a direct, programmatic, and uniform interface to various clinical database implementations. Input-driven systems could acquire data directly from electronic medical records rather than requiring users to enter all patient parameters manually. File-driven systems could delegate certain data-manipulation tasks to database management systems, which are typically optimized to perform complex operations upon large data sets. Database-driven systems could access online databases using a standard API, facilitating their porting to clinical environments that have different database implementations. Databank-driven systems could exploit the query translation mechanism to pool data from disparate sources without laborious manual abstraction. In order to realize these benefits, of course, the abstract query model must be sufficiently powerful to provide decision-support systems with access to all data that they require, and the query-translation mechanism must be sufficiently sound to ensure that data retrieved from different sources have the same meaning.

2.2.1 Clinical practice guidelines

There is a class of the database-driven systems that can benefit particularly from a standard query model and an automated query-translation mechanism. These systems are applications that implement clinical practice guidelines. Clinical practice guidelines (CPGs) are systematically developed statements to guide practitioners' decision making in specific clinical circumstances [Field and Lohr 1992]. The intent of CPGs is to improve the quality and to control the costs of health care by specifying *protocols* for providers to follow in managing particular classes of patients; protocols may include the sequence in which actions should be taken and the clinical criteria for various medical interventions. Protocols can improve the quality of care by recommending medically warranted actions that physicians have overlooked, and protocols can control the costs of care by advising against medically unnecessary or inappropriate actions that physicians have ordered. CPGs are typically formulated by central, authoritative bodies based on analyses of medical literature, and subsequently are made available for distribution to practitioners. For example, the American Medical Association recently published a directory of 1500 CPGs developed by more than 45 physician organizations and other groups [Toepp and Kuznets 1993].

Observers have remarked that computer technology may play an important role in the dissemination and implementation of CPGs by allowing CPGs to be incorporated into the routine information management tasks of physicians. For example, the Committee on Clinical Practice Guidelines of the Institute of Medicine (IOM) has recommended the incorporation of CPGs into computerized decision aids, asserting, "Information and decision support systems are crucial elements in long-term strategies for promoting the application of guidelines. As information and decision-support strategies are used in tandem with other kinds of implementation approaches, they can simultaneously present information and shape and control its uses." [Toepp and Kuznets 1993]. As discussed in Section 1.1.2, however, centrally formulated CPGs can be disseminated conveniently via computer applications only if the applications have *both* the database-querying capabilities needed to establish clinical contexts automatically *and* the portability needed to implement guidelines at a variety of provider sites. The IOM has recognized portability as an important factor in the electronic application of CPGs and has called for standards to permit guidelines to be used on many different types of computer-based equipment and systems. The standards identified by the IOM include "rules of syntax, database structures, and communication links that support multiple settings of care" [Dick and Steen 1991].

2.3 Clinical Data Standards

Consistent with the needs recognized by the IOM, a number of national organizations already have developed or are currently developing standards for the representation of clinical data and decision-support knowledge. These standards bodies include ASTM, HL7, IEEE and ACR/NEMA [McDonald, et al. 1991]. Most standardization initiatives focus on the specification of standard message formats and communication protocols for the electronic interchange of clinical data and on the development of standard programming languages for the electronic interchange of decision-support logic.

2.3.1 Medical data interchange standards

The HL7 message standard [HL-7 Working Group 1989] specifies the syntactic structure of ASCII messages that transmit data regarding medical events, such as admissions, orders, and laboratory results, among clinical information systems. The HL7 standard defines the types of messages that may be transmitted, the records that make up each message type, and the set of data elements in each record. For example, the message specification for "Lab Results" consists of a header record signifying the identity of the patient being tested, followed by multiple detail records, each consisting of the data elements TEST NAME, TEST RESULT, and TEST RESULT UNITS. Although HL7 has been adopted by over 100 commercial vendors as a standard format for the exchange of clinical data, it is strictly a communication protocol and is inadequate as the foundation for a standard query model of clinical information. Specifically, HL7 lacks a *static* conceptualization of clinical data that specifies the set of clinical data objects and the relationships among the data objects in a manner that is independent of any *dynamic* data transmission [McLinden, et al. 1990]. For example, HL7 includes no explicit representation of the relationship between a lab test order and the lab test result corresponding to the order; *HL7 only specifies the syntactic positions of the two objects in a transmitted message*. As a result, the applications that send and receive HL7 messages must be aware of the meanings of and the relationships among data objects, which creates the potential for inconsistent and incorrect interpretations. In order to provide a uniform query interface to clinical information systems, a clinical data standard must explicitly and formally represent the set of data objects and the relationships among the data objects that are represented in clinical database systems; otherwise, there is no way to ensure that communicating applications agree upon the semantics of exchanged data.

The IEEE Medical Data Interchange Standard (MEDIX) [Harrington, et al. 1990], another model for data exchange between automated healthcare systems, addresses the need for a static conceptualization of clinical data by specifying a *reference model* of the health care environment. The MEDIX reference model provides an object-oriented conceptual schema of clinical data objects and their relationships that can be used by applications to confer unique meanings to transmitted messages [Ostler, et al. 1990]. In order to provide a common schema of clinical data across all health care systems, the reference model is specified at an abstract, implementation-independent level. For example, the objects specified in the reference model include patients, physicians, and services provided. It is only within implemented information systems that the conceptual reference model is mapped into specific hardware and software realizations or computational models. The MEDIX approach to defining clinical data standards is compatible with the requirements for application portability that were articulated by the IOM committee on clinical practice guidelines (see Section 2.2.1) because the MEDIX reference model provides a standard, abstract view of the structure and the meaning of clinical data that is independent of specific clinical database implementations. The encapsulation provided by the object-oriented reference model dictates that the implementation of data objects be hidden from client applications and that all operations upon data objects, including queries and updates, be specified by sending messages to abstract objects of the reference model. According to the MEDIX architecture, an "agent process" must exist at each database site that transforms the object-based messages sent to the reference model to non–object-based operations, such as SQL queries, that are supported by local database implementations. The MEDIX architecture represents a powerful and general step toward providing decision-support applications with uniform access to patient data across database implementations. However, before application portability can be realized, there must exist a methodology for conveniently specifying and performing the *transformation* of MEDIX-level operations to local database operations. Such a methodology, which has not been addressed by the MEDIX standards committee, will be instrumental in providing MEDIX interfaces to disparately implemented clinical databases. The development of this methodology is the focus of my research.

2.3.2 Medical knowledge-representation standards

In addition to standards for the representation and communication of clinical data, the development of portable clinical practice guidelines requires standards for the representation of the medical logic that evaluates data. A recent initiative toward standardizing the representation of medical decision-support logic is the Arden Syntax [Hripcsak, et al. 1994]. The Arden Syntax is a programming language that represents medical knowledge in discrete, independent programs called Medical Logic Modules (MLMs), which query a database for relevant clinical data and subsequently process the data in order to generate advice based on encoded medical logic. Typically, MLMs generate alerts, reminders, and critiques for providers based on the automated recognition of noteworthy patterns in a clinical database. The paradigm of pattern recognition and alert generation is appropriate for implementing certain clinical practice guidelines.

Although other languages exist for encoding data-driven medical knowledge [Adams 1986], these languages are specific to the particular clinical computing environments in which they were developed. The Arden Syntax is distinct in that it is specifically designed with portability in mind, that is with the ability to be implemented in numerous, diverse clinical computing environments. Portability is achieved by standardizing the constructs for representing medical logic, such as data types and control structures, and by *underspecifying* the constructs for formulating database queries, such as database structure and query language syntax. The designers of the Arden Syntax intentionally chose to underspecify the query constructs because, after reviewing a number of clinical information systems, they concluded that no single syntax for database queries could be devised that would be general enough to accommodate the diversity in terminology, in database structure, and in query languages that exists among clinical information systems [Hripcsak, et al. 1994]. Rather than attempting to specify a complete query syntax, therefore, the designers of Arden specified a syntax for only those constructs determined to be common to many clinical information systems. These constructs consist of aggregation operators and temporal constraints (see Figure 2.1). The remaining constructs required to completely specify database queries are encoded informally at the time an MLM is authored (usually in natural language), and are implemented in a formal manner that is consistent with the query language, database structure, and medical terminology of a specific institution only later. As a result, the Arden standard specifies the division of queries into two disjoint components: an institution-specific component, which is informally specified and must be *manually* translated into the formal syntax of an institution's local query language, and a more general component, which is formally specified in the Arden Syntax and must be *merged* with the institution-specific component during compilation of MLMs. The syntax and semantics of the query model are described elsewhere [American Society for Testing and Materials 1992].

Figure 2.1 shows a database query formulated in the Arden Syntax. The query is intended to retrieve the name and dosage of the diuretic medication prescribed to a patient most recently within the last week.

```
(drug, dose) := read last { select Drug_name, Drug_dosage
                            from MEDICATIONS
                            where Patient_name = "Williams" AND
                                  Drug_Class = "Diuretic"}
                where it occurred within the past 7 days
```

Figure 2.1: A database query encoded in the Arden Syntax The text appearing within the braces represents the institution-specific component of the query, in this case specified in SQL. The boldfaced text represents the formal Arden constructs of the query, consisting of the aggregation operator “last” and the temporal constraint “where it occurred. . .” (boldfaced for clarity only).

Note that, although the institution-specific component of the query in Figure 2.1 is specified in SQL, this is an arbitrary (albeit convenient) notation. Another author could specify the institution-specific logic of the same query using natural language or machine code. These alternative representations all would conform to the Arden standard, because the institution-specific component of a database query is completely outside the scope of the Arden Syntax. While this feature is an effective short-term expedient to accommodate the diverse database implementations and medical vocabularies in use at various sites, an informal query model actually compromises the portability of MLMs in the long run. Without a standard database schema and query syntax, the query model can have no well-defined semantics. Without precise semantics, database queries are interpreted subjectively at "client" institutions and may be implemented inconsistently when the queries are manually translated. A codeveloper of the Arden Syntax remarked upon this difficulty following a recent experiment in sharing Arden programs:

> "Although standards for representing clinical decision logic can be of great assistance in sharing the work of many, sharing may be delayed until common standards exist not only in the description of the logic, but in all aspects of the medical information system" [Pryor and Hripcsak 1994].

Furthermore, without a standard database schema and query syntax, there can be no formal foundation upon which to build compilers to automatically transform MLM queries to data-manipulation operations that can be executed by diverse clinical information systems. Thus, although the Arden standard provides a uniform syntax for encoding the medical *logic* that makes up clinical practice guidelines, it does not provide a uniform model of medical *data* that would allow MLMs to interface with diverse clinical information systems in a consistent and portable manner. Until standard models of clinical data and of clinical-data retrieval are incorporated into languages for sharing medical logic, the portability envisioned by the IOM for electronic CPGs will not be realized.

As demonstrated by the advent of HL7, MEDIX, and Arden, standards bodies in medical informatics are specifying *parts* of the standard environment for developing portable decision-support applications, but the parts also must be coordinated and integrated before a functioning environment can exist. Specifically, standard languages for encoding decision-support logic, such as the Arden Syntax, must be integrated with standard models for representing and retrieving clinical data, such as the MEDIX reference model. In addition, powerful and sound methodologies must be developed to translate automatically standard representations of database queries into implementation-specific query operations so that diverse databases sites can conveniently and cost-effectively integrate portable decision-support applications into their local environments.

2.4 Contributions of Database and Artificial Intelligence research

Clinical medicine is one of many domains that will benefit from the development of standard representations for enterprise data and decision logic. Accordingly, domain-independent research in the fields of database systems and artificial intelligence (AI) has addressed the sharing of data and knowledge among independent software agents. In the field of database systems, this research has focused on the development of heterogeneous database systems, and in the field of artificial intelligence, this research has focused on the development of sharable knowledge-representation formalisms. It is instructive to review research results from both fields when evaluating the technology available to support portable knowledge-based applications in medicine.

2.4.1 Heterogeneous database systems

Heterogeneous database systems (HDBSs)[1] are computational models for the integration and aggregation of data from disparately implemented databases [Sheth and Larson 1990]. For example, an HDBS might provide uniform access to electronic patient records in a hospital computing environment that uses a MUMPS hierarchical database for storing patient demographic data and a Sybase relational database for storing patient laboratory results. HDBSs are sometimes confused with distributed database systems (DDBSs) because both provide a unified view of and common interface to data that is physically stored in different locations. However, DDBSs are much more integrated and coordinated than are HDBSs. Typically, the constituent databases of a DDBS implement the same data model and query language and run the same distributed database management software. Also, the fragmentation of data in DDBSs is usually *induced* to reap the efficiency and autonomy advantages of distributed computing. The constituent databases in HDBSs, in contrast, existed prior to the establishment of the HDBS and are coordinated much more loosely. The HDBS model, therefore, is more applicable than the DDBS model for providing medical decision-support applications with a uniform interface to disparate clinical

---

[1] The terms "heterogeneous database systems" and "federated databases" are used synonymously in database literature.

databases. Specifically, HDBSs have the following characteristics [Sheth and Larson 1990]:

1. *Representational heterogeneity*. The constituent databases in an HDBS may use different data models, different query languages, different terminologies, and different schema structures to represent the same real-world semantics. In other words, although the data stored at multiple sites may have identical real-world semantics, the data representations and the data-access methods at the sites may differ.

2. *Local autonomy*. Each constituent database has the right to control access to its own data by the HDBS and it has the ability to access and manipulate its own data independently of the HDBS. Most decisions regarding the representation and manipulation of data are made by local database administrators to accommodate local system needs (such as functionality, performance, and cost). The membership of the database in a heterogeneous database system is incidental to the primary purpose of the database.

3. *Bottom-up integration*. Whereas DDBSs *induce* the distribution of data that were previously integrated to achieve efficiency benefits, HDBSs *integrate* data that were previously distributed to achieve interoperability benefits. Bottom-up integration implies that the HDBS must provide an interface to diverse, preexisting information systems without requiring extensive modifications of preexisting software.

These three features describe accurately the situation faced by developers of portable decision-support applications that require access to clinical data at multiple provider sites: Preexisting clinical database implementations are extremely diverse, are maintained to meet local functionality requirements, and are very costly to modify. In this environment, the goal of an HDBS is to provide *database transparency* to portable applications, that is, to provide a global and consistent database interface for applications, as if the data were not distributed and all of the database management systems were of the same type. HDBS research is predicated upon the belief that heterogeneity at the level of constituent database systems will (and should) persist, despite standardization efforts. This certainly is a realistic model for clinical databases in the United States in the foreseeable future.

To provide database transparency in the face of representational heterogeneity, HDBS researchers have focused upon several research issues that are relevant to the development of a uniform query model of clinical information. These issues include:

1. *Common data models and common query languages*. The specification of a single data model to represent the global schema of distributed data and of a single query language to specify site-independent operations upon distributed data are essential to providing database transparency to applications that use HDBSs. Experimental HDBS systems typically have used simple semantic data models and set-oriented query languages, such as the functional model and the DAPLEX query language [Smith, et al. 1986] and the entity–relationship model and the GORDAS query language [Ceri and Pelagatti 1984]. Simple data models that represent "atomic" facts, such as entities and the relationships among entities, are easier to translate to the more complex data models that are implemented by constituent database systems. Set-oriented query languages, such as the relational algebra, provide greater potential for query decomposition and query optimization than record-at-a-time languages, such as CODASYL. I have applied both of these principles in my work.

2. *Schema integration*. Database research on integrating the schemata of constituent database systems into a global schema have yielded valuable insights into the resolution of representational heterogeneities among data in different systems [Agarwal 1991, Ceri and Pelagatti 1984]. Techniques for integrating inconsistent schemas have included the definition of auxiliary data structures to specify mappings across naming conflicts or units-of-measure conflicts; these types of mappings are a necessity for HDBSs in the clinical domain, where numerous competing coding schemes are used to record clinical data.

3. *Query translation*. HDBS researchers also have explored techniques to conveniently and faithfully translate queries formulated in a global

data-manipulation language to equivalent queries formulated in the data-manipulation languages of constituent databases. These techniques typically use mapping information that encodes the correspondence between the global schema and a constituent database schema. For example, this technique has been used in the MULTIBASE query translation facility [Smith, et al. 1986].

4. *Query optimization*. Because the global data model and global data-manipulation language may be more powerful than their counterparts in constituent database systems, additional processing that is external to the local database system may be required to execute global queries. For example, the generalization and specialization constructs available in the functional data model have no counterparts in the relational model used by many clinical database systems. Operations to implemente the generalization/specialization constructs must be specified outside of the database query language and integrated with the queries in an efficient manner. HDBS research has addressed the optimization of such hybrid queries.

Although research in heterogeneous database systems has been ongoing for over a decade and is addressing many of the issues relevant to database interoperability in the medical domain, prominent standardization efforts in medical informatics, such as the MEDIX reference model and the Arden Syntax, have not adopted explicitly the HDBS paradigm or HDBS research results. My own research is based closely upon the results in global data-model development, schema integration, and query translation that originated in HDBS research, as I describe in Section 2.5.

2.4.2 Standards for knowledge representation

Researchers in knowledge-based systems have recently addressed the issues of sharing and reusing domain knowledge among knowledge-based applications to reduce the development costs of such applications [Musen 1992]. The challenges inherent in this objective and the techniques under development to achieve it also are relevant to the task of sharing decision-support applications that require access to clinical databases.

The largest project in this field, the Knowledge-Sharing Effort (KSE), is a collaborative research project to identify and address impediments to the sharing of knowledge among applications [Fikes, et al. 1991]. The specific objective of the effort is to find ways of preserving existing knowledge bases (KBs) so that they can be shared, reused, and built upon by many applications. The central problems addressed by this research are the heterogeneity of knowledge-representation (KR) languages and the heterogeneity of domain terminologies.

The heterogeneity of knowledge representation languages prevents KBs from being shared and reused because the syntax used to represent domain knowledge in one application environment may be completely meaningless within another environment, although the semantics of the represented domain knowledge are similar. The KSE is addressing this problem in two ways: (1) the definition of a *standard* knowledge-representation language, called the Knowledge Interchange Format (KIF) [Genesereth and Fikes 1991], to encode knowledge in an abstract, implementation-independent way; and (2) the development of tools to translate KIF knowledge bases into the knowledge representation languages of various application environments [Fikes, et al. 1991].

As with HDBS research, the work of the KSE is predicated upon the assumption that diversity among knowledge-representation languages will and should persist, which implies that interoperability will depend upon techniques to faithfully translate encoded knowledge among disparate systems.

The second problem addressed by the KSE is the heterogeneity of domain terminologies. Heterogeneity among the terms used to denote and relate objects in the domain of discourse prevents KBs from being shared and reused because different knowledge-based applications may disagree upon the meanings of symbols (terms) that appear in shared KBs; the applications may, therefore, interpret encoded knowledge inconsistently. The inconsistent use of a shared KB implies that at least one application using the KB is interpreting the knowledge incorrectly. Inconsistencies in the interpretation of shared knowledge will occur if applications associate the same term with different meanings or use different terms to denote the same meaning. The KSE is addressing this problem by developing a *standard* language to represent domain terminologies (that is, to represent the set of applicable terms and their meanings) and a method of translating terminologies encoded in the standard language into various KR languages. Also, the KSE is developing a set of tools to verify the consistency between the domain terminology appearing in a shared KB and the domain terminology appearing in an application that will use the shared KB [Fikes, et al. 1991].

Of the two problems being addressed by the KSE, the representation of domain terminologies in a standard, implementation-independent way and the translation of domain terminologies to specific application environments are the most relevant to the objective of providing decision-support applications with a uniform interface to clinical databases. Domain terminologies for knowledge-based systems are quite similar in content and in purpose to conceptual schemata for database systems. Specifically, a domain terminology provides a *conceptualization* of the entities and of the relationships among the entities that may exist in some universe of discourse [Fikes, et al. 1991]. In the field of artificial intelligence, such a conceptualization is called an *ontology*. The standard ontologies envisioned by the KSE define sets of terms that denote classes, relations, functions, and object constants for some domain of discourse. Standard ontologies also include a logical axiomatization to constrain the interpretation of the terms [Gruber 1993]. By this definition, an ontology is similar to a query model in that both consist of the following components:

1. A formal abstract model for representing the properties of objects in a domain

2. A definition of the objects classes and of the relations and functions that may be defined over the members of those classes in a particular domain

3. A specification of the object constants that may be members of the defined object classes

If an ontology also includes a formal query language, then the ontology is equivalent to a query model, as defined in Section 1.2.1.

In the KSE, ontologies are translated from the standard ontology language, which is based on KIF, to the various KR formalisms of implemented knowledge-based environments, such as LOOM [Yen, et al. 1991]. Ostensibly, the translations preserve the semantics of the ontologies. Whereas the KSE is interested in translating between standard domain ontologies encoded in a variant of KIF and various implemented KR formalisms, HDBS researchers are interested in translating between standard conceptual database schemas, such as the MEDIX reference model, and the logical databases schemas of implemented database systems.

Given the similarities in content and in purpose between domain ontologies and conceptual database schemata, at least two results from the KSE's research on ontologies are applicable to the development of environments for portable decision-support applications in medicine: ontology design criteria and ontology translation methods.

Gruber has specified a set of design criteria for ontologies that are intended to support knowledge sharing and interoperability based on a shared conceptualization [Gruber 1993]:

1. **Clarity**. An ontology should effectively communicate the intended semantics to humans who design applications based on the ontology.

2. **Coherence**. An ontology should be internally consistent.

3. **Extensibility**. An ontology should support the addition of new terms (to support the needs of a specific site or when general domain knowledge increases) without requiring revisions of the existing definitions (and, by implication, of applications that use those definitions).

4. **Minimal encoding bias**. An ontology should be specified at a sufficiently abstract level to enable its incorporation into agents that may be implemented in different representation systems and styles of representation.

5. **Minimal ontological commitment**. Because an ontology serves a different purpose from that of a knowledge base, it should include the *minimal* set of domain knowledge to support the intended knowledge-sharing activities, that is, it should make as few claims as necessary about the world being modeled.

The methods used to translate standard ontologies into the specific KR formalisms of implemented knowledge-based systems also may be applicable to the translation of standard database queries into the specific query languages of implemented clinical database systems. The KSE translator operates much like a standard programming language compiler in that it has a *recognition component* to parse sentences based on a specification of the (standard) input grammar and a *generation component* to generate translated ontological statements from the internal representation of the input stream. Because a transformation must take place between the input and output ontologies, the KSE translator also has a *reformulation component* that applies rewrite rules to an internal representation of the parsed input to create a new internal representation that is consistent with the representational formalism of the desired output [Fikes, et al. 1991]. The reformulation component is a specialized type of theorem prover that derives output statements that are *equivalent* to the input statements, rather than implied by them. In this sense, it is comparable to standard rewrite systems, such as attribute grammars [Knuth 1968], that apply semantic rules to generate translated outputs. The reformulation component is distinct from attribute-grammar methods, however, in that it is goal-directed and applies heuristics to guide the choice of rewrite rules. The reformulation process, therefore, entails a search of the space of possible transformations to effect the desired translation, with possible backtracking. This model of ontology translation is also used by the SIMS system [Arens, et al. 1993], a prototype heterogeneous database system that applies rule-based techniques to integrate database schemas and perform query translation. Attribute grammars, conversely, support the *deterministic* modeling of transformation rules such that the rules always are applied in a predetermined sequence, which avoids heuristic search and backtracking. I have chosen attribute grammars as the formalism for representing and executing the translation of queries because deterministic modeling renders the translation process more efficient. However, it may be instructive later to compare the efficiency and effectiveness of the attribute-grammar and rule-based models for rewriting queries, and it may be useful to apply rule-based methods to optimize translated queries.

2.4.3 Combining database and knowledge-based research

Although database and AI researchers often have different perspectives on similar problems, the challenge of providing decision-support applications with a uniform interface to diverse database systems constitutes a problem that applies to both research fields. The fundamental basis for this duality is the fact that database systems *are* knowledge-based systems: Database systems represent knowledge about the world by implementing some computational model that supports both the storage of data items and the description of the stored items. This definition applies equally well to what AI researchers call domain ontologies. Sowa asserts that the primary difference between knowledge-based and database systems is the ratio of the number of data descriptors to the number of stored data items: database systems store many repetitions of a small number of simple data types, whereas AI systems have unique and complex descriptions for almost every data item [Sowa 1984] Of course, AI systems also provide complex inference techniques that allow data processing far more sophisticated than that supported by database query engines, but the inferences performed by knowledge-based and database systems differ more in sophistication than in nature; both types of systems provide information based on the value-added processing of data.

This philosophy has motivated the development of software modules known as *mediators* [Wiederhold 1992]. Mediators apply database and knowledge-based techniques to provide interfaces between applications and sets of autonomous databases. Mediators apply database techniques to locate, retrieve, and integrate data from heterogeneous database systems, and mediators apply knowledge-based techniques to perform value-added abstractions and simulations based on the retrieved data. One of the primary purposes of mediators is to insulate applications from the complex and changing features of the databases that they access. By creating a single uniform interface to multiple heterogeneous information sources, a mediator greatly simplifies the development of applications that need to access those data sources.

Given the similarities between database and knowledge-based systems at a fundamental level, the strategy of applying both database and knowledge-based techniques to mediate between applications and databases is appropriate to the current objective of interoperability in medical informatics. For example, the principles that AI researchers are discovering for the specification of robust sharable ontologies and the reasoning tools that AI researchers are developing for the validation of domain ontologies certainly are applicable to the development of a standard conceptual schema of clinical data, such as the MEDIX reference model. Also, the techniques developed by database researchers for the resolution of semantic discrepancies among database schemata and for the optimization of queries that are distributed among heterogeneous systems certainly are applicable to the development and implementation of portable languages for the representation of decision-support logic, such as the Arden Syntax. Because the task of providing medical decision-support applications with access to patient data that is stored in disparate clinical information systems is a necessarily complex undertaking, sound research results from all applicable fields should be brought to bear upon it. The result of this approach will be a *clinical information mediator* that applies database and AI techniques to insulate CDS applications from the heterogeneities and complexities of clinical database systems.

2.5 Components of the Solution

The technology that I propose to provide decision-support applications with access to patient data in heterogeneous clinical information systems consists of a standard abstract query model of clinical information and a means to translate standard abstract queries to implementation-specific operations. Previous research exists relating to each component of this strategy: An abstract data model and corresponding query language, a standard schema of clinical information represented in the abstract data model, and a means of translating queries against the standard schema to queries that can execute against diverse clinical databases. My work is based upon certain results of this past research.

2.5.1 An abstract data model and query language

As shown by HDBS research, database interoperability requires the use of a common data model that is sufficiently simple and abstract to translate conveniently to various implemented data models and a corresponding query language that can be used to formulate queries at an equally abstract level. *Semantic data models* [Hull and King 1987, Peckham and Maryanski 1988] provide appropriately abstract conceptualizations of domain data that can serve as common modeling environments for disparately implemented databases. Although semantic data models (SDMs) have been traditionally used for database design because they provide more powerful abstractions for the specification of database schemas than are supported by the relational, hierarchical, and network models, the abstract and expressive modeling constructs of SDMs also are well suited for the specification of conceptual schemas (see Section 2.1) that model domain data in an implementation-independent way. Most SDMs provide the following abstractions:

1. **Classification**: The designation of individual objects as members of an abstract object class. For example, the classification of several persons as members of the class "Patient."

2. **Aggregation**: The designation of multiple object classes as components of a more abstract object class. For example, the aggregation of the classes "Sodium test result" and "Potassium test result" to form the class "Electrolyte test results."

3. **Generalization**: The designation of one object class as the supertype of another object class. For example, the generalization that the object class "person" is a supertype of the object classes "Patient" and "Physician."

4. **Association**: The designation of individual objects as participants in an abstract object that denotes a relationship. For example, the association of two persons such that one is the physician of the other.

The best-known examples of SDMs are the entity–relationshp (ER) model [Chen 1976], including numerous variations of it [Elmasri, et al. 1985, Engels, et al. 1992], and the functional data model [Shipman 1981, Wilkinson, et al. 1989]. Other SDMs include extended relational models [Codd 1979, Elmasri and Wiederhold 1979], and data models based on semantic networks [Mylopoulos, et al. 1980, Sowa 1984]. I have based the conceptual model that I use on both the ER model and the functional model, combining the desirable features of each. I describe the features of this hybrid data model in Chapter 3.

The SDMs suited for supporting database interoperability have corresponding high-level query languages, because a query language is necessary to provide an operational interface to a variety of database systems. The functional data model and the entity–relationshp data model used in the HDBS research projects described in Section 2.4.1 both provide corresponding query languages and were likely selected as common data models for that reason. A third candidate is the conceptual graph (CG) data model, which also has its own query language. Although CGs have not yet been used as the conceptual data model in experimental heterogeneous database systems, they were recently approved by an ANSI standards committee as the normative modeling language for the Information Resource Dictionary System (IRDS) [ANSI-X3H4 1993]. The IRDS is a modeling environment that describes and manages an organization's information system resources and supports the integration of an enterprise's information environment and information models. As such, the IRDS separates the conceptual view of information (its *meaning*) from the way in which information is represented, stored, and processed, so that it can be used and applied in many different ways. Because the use of CGs in this capacity is quite new and yet to be proven, I have chosen to base my abstract query model on the entity–relationshp model and the functional model.

2.5.2 A Domain schema for clinical medicine

The specification of a standard query model for clinical information requires that a standard *schema* of medical data be specified using the chosen conceptual data modeling environment. Although numerous groups are currently pursuing the specification of a standard conceptual schema for clinical data, no such schema has yet emerged in the public domain. The MEDIX reference model described in Section 2.3.1 has yet to be published as a national standard. Although the Computerized Patient Record Institute (CPRI) has called for the specification of a composite clinical data dictionary to support clinical database interoperability, the CPRI is currently developing no such reference model of its own. An independent group of medical informatics researchers recently convened to begin work on a “standard canonical scheme of medical concept representations” [Evans, et al. 1994], but the work of the “Canon” group is still very nascent and is primarily oriented towards standardizing medical terminology rather than medical database schemas. Numerous proprietary efforts to define global conceptual schemas for clinical data also are under way, but these are not likely to be useful in supporting widespread interoperability among clinical databases unless they are published and accepted as public-domain standards. In short, there currently exists no comprehensive and publicly available schema of clinical information encoded in *any* modeling environment that could serve as the global schema in a standard query model for clinical medicine. The definition of a global schema for sharing decision-support applications, therefore, will have to proceed de novo; however, the process will be unhindered by the preexistence of a global schema that has been widely adopted but may be inadequate for the purposes of sharing applications (because it violates Gruber's criteria for shared ontologies, for example).

2.5.3 A query-translation methodology

An abstract data model and a standard global schema of clinical data are only part of the solution for providing applications with a uniform interface to clinical databases. The other part involves "operationalizing" the global schema such that queries issued against it can retrieve patient data that is physically stored in heterogeneous database systems. Unless the implementation of clinical databases is standardized, which is unlikely and may be undesirable, some transformation must take place between the abstract query model against which the queries of shared applications are formulated and the local query models in which clinical data actually are stored. The transformation may take one of two forms: data translation or query translation. Although I have chosen to pursue the strategy of query translation, I discuss both strategies for purposes of comparison.

2.5.3.1 User-mediated translation

In order to appreciate at a higher level of abstraction the data-translation and query-translation strategies that I will describe, it is instructive to consider how shared applications that require patient data achieve widespread portability today. The prevailing model entails the *manual* translation of patient data and of application queries by end users. In this paradigm, used by decision-support systems like MYCIN and QMR, the "global schema" of clinical information is the informal mental model of medicine shared by all professionally trained users. Applications issue queries against this "schema" by formulating questions in the corresponding "query language," English (presumed to be supported at every site). The "query engine" consists of a human end user, who services queries in one of two ways. If the requested data have already been translated by the user from its original representation in the patient chart to its “global conceptual” representation in the user's mind (that is, if the user “knows” the information), the query engine can respond to the query directly and immediately. This mode of query response involves **data translation** from the stored data format (the patient chart) to the shared data format (the user's mental model) prior to consideration of any queries.

Alternatively, if the data is not already known to the user, the English query must be translated into a search strategy against the patient chart to locate the data and to respond to the query. This mode of query response involves **query translation** from the shared formulation (an English question) to a formulation relevant to the stored format (a search procedure in the chart). These two methods of manually servicing portable queries are analogous to the data-translation and query-translation strategies that must be formalized in a computational model to automatically service queries with minimal human intervention.

2.5.3.2 Translation of data

Data translation involves the transformation of data from the various native formats in which they may be collected and stored to a common shared format in which they can be queried by portable applications. Data translation may take place manually or automatically. Manual data translation requires medical records personnel to manually abstract patient records into a format consistent with a standard clinical data set and with a standard coding scheme. The data that are manually “normalized” in this way are subsequently entered into a standard centralized database and all applications that require clinical data query this database. This is the model that many large clinical data banks, such as ARAMIS, employ for aggregating data from multiple sites. Although this method could be applied to normalize data into a standard format that portable applications can query, the costs and time requirements of manually translating each collected datum from a stored format to the shared format are too great to accommodate many decision-support applications.

An alternative method for translating data is to develop computer programs (often called “geteways”) that algorithmically translate data from the stored format in which they are captured to a shared format in which they can be accessed by portable applications. Gateways may convert data directly to the shared format [Marrs, et al. 1994], or they may translate the data first to an "interchange" format (such as HL7 [Rishel 1989]) that is standard across database implementations and that subsequently can be translated to the shared format. Although automating data translation reduces the costs and delay of translating data, this strategy still suffers from two problems. First, data translation entails the duplicate storage of data in both the original and the shared format, which increases both the cost of operating an information system and the chances of compromising data integrity. Second, the correspondence between the stored format and the shared format is represented only in the encoded algorithms of the translating programs, and this "procedural" representation is difficult to inspect, validate, and maintain. A procedural representation of the correspondence between disparate data formats increases the chances of incorrectly translating complex medical data and increases the costs of updating the translation algorithms.

2.5.6.3 Translation of queries

The alternative strategy for providing a common query interface to disparately stored data is to translate the queries rather than the data. This model entails the translation of queries that are issued against the shared format to equivalent queries that can be executed against the stored format. Again, query translation can be implemented either manually or automatically.

Manual query translation requires that database programmers reformulate each query represented in a portable application into an equivalent query that is consistent with the database system at their provider site. The programmers perform the translation based on their interpretation of the meaning of the original query and on their understanding of the meaning of the locally stored data. Manual query translation is the method proposed for sharing database queries that appear in medical logic modules encoded in the Arden Syntax [Hripcsak, et al. 1990]. The problems inherent in this method of query translation include the effort required to perform the translation and the potential for semantic errors introduced by the translation. Because there is no formal standard model of clinical data and because the structure of clinical databases is complex and highly variable, considerable effort is required to understand the meaning of a shared query and to implement the same meaning with respect to a local database environment [Hripcsak 1994, Pryor and Hripcsak 1994].

Automated translation of queries provides, in theory, the opportunity to remedy the deficiencies of manual query translation. Automated query translation requires that programmers represent only a *mapping* between the shared data format and their local data format. A query-translation and query-execution *engine* uses the mapping to automatically reformulate each query into an equivalent local query and to execute the local query. This strategy requires less manual effort and allows the systematic representation and application of mappings between query models. I base my approach to providing a uniform query interface to disparate clinical databases on this strategy, and I discuss two variations of the strategy that appear in current research systems and that have influenced my work.

1. *Automated query translation based on canonical mappings*. In this method, an entity–relationshp (ER) schema initially is designed and then algorithmically transformed into a corresponding relational schema, based on a unique and canonical set of schema transformations. Queries issued against the ER schema are automatically translated to equivalent relational queries by a compiler that performs query transformations based on the canonical mappings. Essentially, the mapping between any ER schema and its corresponding relational schema is *hard-coded* into the translating compiler. This method is used by two independent research groups to translate queries from an ER-based query language to an equivalent relational form [Hohenstein 1990, Markowitz and Shoshani 1993]. The advantage of automatically translating queries based on *canonical* mappings is that the mappings and the corresponding query-translation algorithm may be readily formalized and validated because they are invariable, regardless of the specification of the ER schema. One of the research groups [Hohenstein and Engels 1990] formally models the translation algorithm as an attribute grammar [Knuth 1968], which constitutes a declarative representation of the translation algorithm that can be formalized and validated.

The disadvantage of query translation based on canonical mappings between an ER schema and a relational schema is that it precludes the use of a single ER schema as an interface to different legacy databases; the structure of the legacy databases will not conform, in most cases, to the canonical structure required by the query-translation algorithm. The missing element is a flexible means of representing the mapping between a conceptual schema and a relational schema, such that the mapping may be both validated and used to drive query translations. Although I have borrowed the technique of representing mappings in an attribute grammar, I have enhanced it to be more flexible and to support various relational implementations of a single conceptual schema.

2. *Automated query translation based on procedural mappings*. In this method, the mappings between the conceptual database schema and various underlying database implementations are represented as procedural functions that physically import objects stored in underlying databases into corresponding objects stored in the conceptual environment, where they may be manipulated further as part of query processing. This approach characterizes the database-integration strategy of the Physician Workstation project at Hewlett-Packard Laboratories [Annevelink, et al. 1992]. This strategy is effectively a hybrid of the data-translation and the query-translation strategies: Procedural functions translate and import data from a stored format into the shared format, but the functions are only invoked when queries require data that the functions provide. Also, the mapping functions may specify that the underlying database not only retrieve requested objects, but also apply certain of the data operations included in conceptual-level queries (such as filtering results). Currently, the scope of operations that can be specified in this way is limited.

The advantage of query translation based on procedural mappings as opposed to canonical mappings is that procedurally specified mappings can accommodate a wide range of legacy database implementations and are not confined to "canonically" implemented versions of the conceptual schema. The disadvantage of this approach is that the query engine cannot perform certain optimizations because the mappings between query models are specified as *procedural* functions that cannot be decomposed, reordered, and recombined to achieve maximum efficiency [Annevelink, et al. 1992]. To maximize the potential for optimization, a *declarative* representation of query-model mappings is necessary. Because a query optimizer can inspect and manipulate declarative mappings based on an encoded understanding of the operations within the mappings, the optimizer can apply information about the semantics of the data, the state of the database, and the capabilities of the DBMS to determine an optimal sequence and allocation of the operations.

Although experimental methods exist for automatically translating queries from a semantic data model to a relational data model, none of these methods can map a single schema of clinical information to various legacy databases and use the mappings to generate optimized queries. Because portable decision-support applications require these features, I have borrowed from and enhanced the existing methods to develop a model of query translation that can deliver the needed functionality. I describe this methodology in the following two chapters.