Database Modeling and .NET Web Services in the Context of a Distributed Web-Based Hospital Information System

by

Glorimar Ripoll

B.S. Management Science
Massachusetts Institute of Technology, 2001

SUBMITTED TO THE DEPARTMENT OF CIVIL AND ENVIRONMENTAL ENGINEERING
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF ENGINEERING
IN CIVIL AND ENVIRONMENTAL ENGINEERING

AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

JUNE 2002

Signature of Author: .....
Department of Civil and Environmental Engineering
May 10, 2002

Certified by: .....
John R. Williams
Associate Professor of Civil and Environmental Engineering
Thesis Supervisor

Accepted by: .....
Oral Buyukozturk
Chairman, Departmental Committee on Graduate Studies

Database Modeling and .NET Web Services in the Context of a Distributed Web-Based Hospital Information System

by

Glorimar Ripoll

Submitted to the Department of Civil and Environmental Engineering on
May 10, 2002
in Partial Fulfillment of the Requirements for the
Degree of Master of Engineering in Civil and Environmental Engineering

ABSTRACT

The E-Health system offers personalized real-time web-based portals for different players of the health care industry, focusing on the patient-physician relationship. The system does not attempt to substitute personal physician interventions. On the contrary, it encourages health education and more interaction by providing a personalized, secure, easily accessible, and data-rich communication environment. E-Health presents a comprehensive view of the health care practice to key stakeholders, and enhances the way medical information is structured, stored, managed, and delivered.

The health care industry is complex, and such system must effectively address this complexity. This document discusses the system's value proposition, technologies, business model, infrastructure, design, and implementation. Particularly, the discussion focuses on relational database modeling and XML Web Services using Microsoft's .NET Framework. The principal design challenge is creating a flexible database model that captures the needs of the health care world and its multiple players. This challenge requires having a clear vision of the modeled world and how the system will offer value for stakeholders. Other challenges include providing the capability of managing and transferring data for a real-time system. Web Services are used to facilitate database interactions and integration from distributed sources. These services can be easily referenced and invoked from different types of client applications, encouraging code reusability and separation of data management from the presentation layer.

Thesis Supervisor: John R. Williams
Title: Associate Professor of Civil and Environmental Engineering

Acknowledgements

Thanks to my team members for making this project a learning and fun experience: Sakda Chaiworawitkul, Franny Hsiao, Andrew Ferriere, Tashan Yen, and Osamu Uehara. Professor John Williams's advice as project supervisor is appreciated. Thanks for introducing us to the technologies we used to build the E-Health System.

Thanks to my family for their constant and unconditional love. Thanks to my father, Miguel Ripoll, for his financial and moral support, which made my MIT education possible. Thanks to my mother Gloria for her advice and for caring so much about my well being through stressful times. And a special thanks to her friends, who persistently prayed for me and my success and happiness as an MIT student.

Thanks to Luis Alejandro Benitez for being my sunshine at MIT. I met Luis five years ago as MIT freshmen, and since that day, the special bond we share gave me the day-to-day strength I needed to successfully complete my education. His love and support made my MIT experience truly amazing.

Above all, thanks to God for my family and friends, and for giving me the opportunity to experience MIT's world-class education. I learned so much beyond courses and textbooks; most importantly, I learned to push myself to the limits to see what I am capable of. God has never abandoned me and was always by my side giving me the emotional and physical determination I needed.

MIT, thanks for the five most challenging years in my life. I will miss you!

Table of Contents

Table of Figures

1 E-Health Information System

On September 2001, six Information Technology graduate students at the Massachusetts Institute of Technology were brainstorming and evaluating different options for their Master of Engineering nine-month project. The students had recently been introduced to web-based technologies that could be employed using Microsoft's .NET Platform. The technologies were certainly attractive, and the students decided to develop a system based on them. However, they had to first discover a real need or problem to solve.

After an initial market research phase, the group concluded that the health care industry was an excellent choice. The industry is experiencing difficult times, but also witnessing exciting new trends. There are many players in the industry and the complexity of industry processes and rules is evident. However, no hospital was offering their stakeholders a comprehensive system for them to easily manage this complexity. The market offering gap was evident and the industry conditions where just right. This is how E-Health, the next generation of health care, was born.

This document discusses the health care industry background, and the E-Health value proposition, business model, technologies, infrastructure, system design, and development. It focuses on relational database modeling and Web Services development using Microsoft's .NET Framework. In the complex and data-rich health care world, database design and manipulations acquire a special importance.

2 Health Care Industry Trends

The exponentially growing health care industry is witnessing significant trends that are reshaping the industry's future. Key trends include a rising wave of patient

empowerment, and the introduction of modern information technology (IT) tools that improve diagnosis and treatment. Patients desire to have greater involvement in medical interactions and decisions. Thus, they are actively searching online for health related information. In addition, patients are very receptive to new technologies that facilitate interaction with health care professionals (Newman). This trend facilitates the adoption of a web-based system such as E-Health.

Existing web-based services are also addressing the patient empowerment issue. However, the Internet is full of unreliable medical sources that attempt to substitute personal physician interventions with distant online advice (Ramirez). This frightening trend is certainly unacceptable because nothing should substitute the richness of the patient-physician relationship. A valuable system must not attempt to substitute the relationship, but must encourage it.

Other significant industry issues are the escalating costs and the reduction in the quality of patient care. The Health Care Financing Administration affirms that health care spending in the United States exceeded $1.2 trillion in 1999 (Walker). According to the Bureau of Labor Statistics, the federal government's medical expenditure is projected to consume 24.5% of the federal budget by 2008 (Saunders). To reduce costs, some institutions introduced incentives to minimize cost-intensive services, without considering the effects on patient care. This superficial solution to the cost problem had a destructive effect on the quality of care.

Some traditional and modern health care players have been successful at reducing costs and improving quality of care. On the other hand, leaders of the health care consulting practice expressed that "no company has yet assembled anything that

resembles a full platform of capabilities" (Matheson). Any solution that attempts to offer a full platform of capabilities must successfully address the complexity of the health care practice and the recent trends. Realizing that the industry is far from experimenting the full potential IT represents, the E-Health team was determined to revolutionize health care with the use of cutting-edge technologies.

3 E-Health Value Proposition

The E-Health system presents a comprehensive view of the health care practice to key stakeholders. It enhances the way medical information is structured, stored, managed, and delivered. Furthermore, it encourages health education and greater interaction between patients and caregivers. Particularly, the E-Health system offers personalized web-based portals to fortify the patient-physician relationship. This is done by providing a secure, easily accessible, and data-rich communication environment. The personalized portals allow stakeholders to easily manage the information they need to accomplish their respective roles. Each user sees a different slice of information depending on their role. A user's role is determined through the login process.

All the information available through the E-Health system is stored and retrieved from the database servers in real-time. This means that any data changes are instantly reflected in the database. Therefore, all users can access the most recent and accurate information available at all times with no delays. Most importantly, this process does not involve human support, and users can access the application from any Web-enabled device, including wireless PDAs and mobile phones.

Through their personal portals, the different groups of users can complete a variety of activities designed for their particular roles. For example, patients can electronically:

- Enter and track different health measurements
- View personal prescriptions' information
- View personal medical/lab test orders
- Schedule appointments with primary physician
- Send emails to primary physician
- View their medical history
- Update contact information
- Receive alerts if a health measurement exceeds an established critical value

In the case of physicians, they can electronically:

- Access patients' records, including information from other doctors, if they have permission
- Order prescriptions, lab tests, and medical test through a simple interface that confirms the order submission
- Receive alerts of significant changes in patient's vitals signs and other health measurements
- Manage patients' appointments

From the commodity of their home or anywhere else with a wireless device, patients and physicians can access E-Health. Instant access to the information from anywhere eliminates searching time imposed by traditional paper-based records. Electronic records prevent transcribing errors, increase efficiency, and improve

workflow. Moreover, the system provides a precise and current picture of each patient's conditions and risk level. By allowing patients to directly enter health measurements to the system, patients' records contain information that might have not been collected otherwise. These benefits allow physicians to offer a timely health care response.

While maintaining the flexibility needed in the health care world, E-Health helps standardize the medicine practice by providing a common medium to organize and communicate data in the same format. Through the functionalities offered by E-Health, health care institutions can unquestionably witness benefits of cost reduction and higher patient satisfaction.

4 Internet-Centric Technologies

The benefit proposition of the E-Health system is possible because of advances in integration technologies for distributed systems. The .NET Platform is an example of such advances. It enables the seamless integration of numerous internet-centric programming languages, classes, frameworks and protocols, including C#, XML, ADO.NET, Web Services, SOAP, ASP.NET, and WSDL. These technologies are discussed in the following sections.

4.1 .NET Framework

Microsoft's .NET Platform is an innovative software development framework. It provides a novel application programming interface to easily build, deploy, and run scalable applications and services that target any browser or device. The Platform includes one of the largest class libraries available, known as the Framework Class Library. The Library assists in the creation of powerful desktop, client/server, and Web Services applications. Visual Studio .NET is a central tool of the Platform that offers the

development environment and programming interface where this diversity of applications can be created.

4.2 C# (C Sharp)

C# is a relatively new object-oriented language developed by Microsoft to facilitate Internet-centric, high-performance, distributed .NET applications. It was built on other languages such as C++ and Java, thus has obvious similarities to these languages. C# also provides a Rapid Application Development environment, similar to Visual Basic, facilitating speedier implementations. The .NET Framework supports a diversity of existing and new programming languages. Nevertheless, the C# language is probably the best choice for .NET developments. C# was a logical choice for E-Health because it was going to be entirely developed in the .NET environment.

4.3 Extensible Markup Language

The Extensible Markup Language (XML) is at the center of new .NET database technologies and Web Services, thus knowledge of XML is necessary for the development of the E-Health system. XML is quickly emerging as the dominant standard for structuring, transmitting, and exchanging data in the World Wide Web via the Hypertext Transfer Protocol (HTTP). It is not a stand-alone technology, but part of a family of growing technologies and frameworks for data exchange between organizations.

Because XML is technology and platform independent, it makes possible the communication and data integration between disparate systems. Thus, it is becoming increasingly popular for allowing integration with legacy systems. This means that any system is able to receive and interpret the XML document without worrying about the

source's platform, and vice versa. For example, the source could be .NET-based, and the receiving client could be a desktop application, a Java-based client, a mobile device like an iPAQ, an Oracle database, or an ASP.NET system like the E-Health Web Application.

Similar to the Hypertext Markup Language (HTML), XML is a tag-based markup language. Yet, contrary to HTML, it allows user-defined tags that make XML documents more flexible. The major objective is to organize information in a way that human beings can comprehend it and computers can technically interpret it.

4.4 ADO.NET Database Technology

Databases are used to store and manage data from organizations. Although data storage and manipulation technologies have been around for many years, transferring data from one organization to another was not a simple task. With the advent of XML, there is a common medium for exchanging data among organizations independently of the source and destination platforms.

The .NET Framework Class Library includes a collection of classes to manage database interactions, which are known as ADO.NET. ADO.NET is fundamentally based on data handling methodologies such as DataSet. A DataSet is a class that has a collection of methods to process XML-based data. DataSet objects can store complex information and relationships in a single, portable, and structured entity. This is possible because the objects enclose multiple DataTable and DataRelation objects.

DataSets can be populated dynamically with the results of a database query or from an XML documents. The DataSet can be locally modified, and later synchronized with the backend database using update operations. Synchronization is feasible because DataSets were designed to be disconnected objects. The DataSet could also be

transformed back to an XML format. This transformation is possible because the DataSet is based on XML, and thus, retains the XML structure.

XML is so tightly integrated in the .NET Framework that HTTP data transfer and manipulation employing .NET XML-based objects is simple. These characteristics make XML-based DataSets a great choice for encapsulating and transferring data between E-Health clients and the Web Services.

4.5 Web Services

Corporations can think of Web services as business functions carried over the Web. Technically, Web Services are objects and methods that can be invoked from any client over HTTP. Unquestionably, these are a central piece of the .NET Framework, and once again, XML support is fundamental in Web Service technologies. The framework provides built-in support for invoking Web services without the need of additional tools. Unlike previous distributed-systems technologies, Web Services are constructed on the Simple Object Access Protocol (SOAP). SOAP uses the XML standard to describe data and allows sending and receiving messages through HTTP.

Because it is based on industry standards, different client platforms can access Web Services by calling remote methods using SOAP XML-based messages. The Web methods return the resulting data types and objects represented as another SOAP message. This enables primitive types, such as integers and strings, and complex DataSets to be transmitted over the Web as XML. In the case of DataSets, the data returned contains two sections. The first is an embedded XML schema describing a particular database table, and the second contains all the retrieved table records. Being

able to return DataSets is one of the most powerful uses of Web Services. In fact, most E-Health Web Service methods return DataSets.

A Web method must have a definition like any other non-Web method. This includes the method's name and its parameters with their corresponding types. The method definition must also indicate the access modifier of the method. For Web methods, the access modifier must be public so that external classes can access them. The method's return type must also be specified in the definition. Finally, to make the method accessible as a Web method, it needs the [Web Method] attribute. Web methods that access databases must create a database connection object, and set the connection string of the database that needs to be accessed.

Many firms are exploring Web Services to do a range of tasks from sharing information with internal computer systems to tightening links with business partners. A given Web Service can perform functions in multiple applications freeing up time and resources by promoting code reusability and efficient development. The distributed nature of Web Services makes this possible allowing separation of the interface from computing and database functions. As it will be evident in future sections, Web Services also contribute to the creation of powerful new business models.

4.6 ASP.NET Web Applications

ASP.NET is a revolutionary programming framework that enables the rapid development of powerful applications for different clients. It is based on Microsoft's original Active Server Page (ASP) framework. In simple terms, ASP is a specification for a dynamically created Web page. When a browser requests an ASP page, the Web server generates a page with HTML code and sends it back to the browser.

All ASP.NET Web pages have the .aspx file extension. The .aspx file represents the graphical application interface. The actual code behind the application is located in an .aspx.cs file. The .cs extension represents a C# document. Other ASP.NET compatible languages have different file extension after the .aspx specification.

The original ASP model underwent major modifications to include new functionalities that are the foundations of ASP.NET. The ability to develop and access remote Web Services is probably the key feature of the new ASP framework. Web Services can perform functions needed by ASP.NET Web applications, freeing the application of maintaining all the functions locally. In the ASP.NET environment, Web Services are coded in a .cs.asmx file. The .asmx file provides a dynamically created interface to test and access the services through a browser.

ASP.NET allows straightforward integration of Web interfaces and services. The two modules need to be compiled together, but first, the Web application must reference the services. To do this, it is necessary to create a Web Service proxy class. If the Web address of the service is known, as it is the case with E-Health, the process is as simple as typing the Web Service URL in a special dialog box. It is also possible to search the Universal Description, Discovery and Integration (UDDI) directory for other services. UDDI facilitates the search for Web Services offered by other companies that might match your needs.

When the URL is submitted, a dialog window shows the service documentation page, including the Web Service Description Language (WSDL) contract. WSDL is an XML document that defines the methods, parameters, and data types of the Web methods available in a Web Service. By accepting the contract, a WSDL proxy is created. The

proxy class represents the service on the client side and makes the Web Service simple to use. The client application will interact with this proxy instead of with the service directly by wrapping the requests and return messages into SOAP. With the proxy, the Web methods are invoked just like any other local class methods.

ASP.NET also offers a series of innovative server controls. One example is the DataGrid control, which can be easily filled with the information contained in a DataSet object. When a Web method makes a database retrieval, the information is taken out of the database server and is transferred as XML through HTTP. ASP.NET receives the DataSet, which is an XML database representation including the specified table records. The application recognizes that the XML file is representing a DataSet object. Thus, the file can be easily loaded into a DataGrid because this control connects to a DataSet object. Actually, the DataGrid is bounded to a specific DataTable within the DataSet returned by the Web method. Finally, the ASP.NET control displays the information as HTML.

Besides the DataGrid control, other Web controls allow DataSet binding and permit specifying the columns that the application displays. This functionality is very important for the development efficiency of the E-Health application. For example, if the application needs to display a patient's name and his address in two different controls, it is not necessary to perform two specialized queries and Web methods calls (i.e., one that returns the name, and another that returns the address). Only one query that returns all patient information in a DataSet object is needed. Once the DataSet is received by the application, each control can select different columns of the DataTable.

Finally, ASP.NET offers excellent tools for dealing with events. Events are fired when a user clicks on a button or makes a selection from a list, for example. The application can handle the events locally or through Web Services calls. The E-Health application will primarily use the latter to deal with its events.

5 E-Health Business Model

During a brainstorming session, the E-Health development team discussed different business models for the system. For the purpose of prototype demonstration, the team selected a localized-independent model. This means that each hospital has its own database, and accesses the E-Health application and services independently of any other hospital. The prototype design that is discussed in the rest of this document is based on this local hospital model.

The.NET technologies and XML industry standards open the door to a new world of business models. It is important for the development team to explore and be aware of these possibilities. This section discusses a new business model that takes advantages of the distributed nature of the .NET technologies. The model captures the potential impact that E-Health, using cutting-edge integration technologies, can have on the health care industry.

The proposed business model is a modified version of the well-known Application Service Provider (ASP) model that has been implemented by numerous software firms. ASPs are third-party entities that manage and distribute software-based services and solutions to customers across a wide area network from a central data center. A vertical market ASP provides support to a specific industry, such as healthcare. In

essence, ASPs are a way for companies to outsource some or all aspects of their information technology needs.

Built on the vertical market ASP model, the proposed E-Health business model is an Application-Web Services Network, or E-Health Network (EHN). A health care institution that joins the EHN can have access to the E-Health Web application and its related Web Services. The Network encourages communication between affiliated institutions by facilitating knowledge and resource sharing. The EHN model also facilitates data mining services to create new knowledge within the Network community.

The EHN targets the needs of wide array of institutions including hospitals, pharmacies, laboratories, medical testing facilities, and insurance companies. Being a web-based network, institutions can access EHN from any computer or device that connects to the Internet. There is no need for sophisticated software or hardware installation and maintenance. Affiliated institutions would have to pay a fee for the use of the application and services, plus a minimal fee to connect to the Internet. Patients that are affiliated to a hospital belonging to the EHN will have access to personal portals and other EHN resources at no cost beyond the Internet connection fee.

The EHN supports distributed and local data centers. Institutions may decide to keep the data locally or outsource database hosting to the Network. Besides institution-specific databases, the EHN stores a group of database tables that are common for all institutions. Common data includes drug information, lab test procedures, and health condition information. In other words, the EHN will manage all validation tables by keeping them up-to-date.

In essence, the Network uses revolutionary technologies to encourage constant communication and interaction among different industry players, including the patient. This communication is maintained through the personalized, secure, accessible, and data-rich environment that the EHN represents.

6 E-Health Technology Infrastructure

The technologies described in previous sections must be organized intelligently to provide the technical infrastructure needed to support any business model, including the E-Health Network. Figure 1 depicts the infrastructure design. Although the E-Health prototype focuses on patients and physicians, the infrastructure design allows a wide variety of user groups to access E-Health. These include patients, physicians, nurses, paramedics, lab technicians, pharmacists, hospital administrators, and insurance companies. These users can utilize different client devices to connect to the application. For example, they can use the browser in a PC/Workstation, a laptop, a wireless PDA, or a mobile phone. All devices are able to connect to the application through the Internet's public and existing network infrastructure. A Web server accepts and manages requests for the different clients, and routes them to the appropriate location. The ASP.NET application sitting on the Web server invokes Web Services server methods when necessary. All the data transfer between these two servers is done entirely through HTTP over the Internet. Web Services can access the E-Health local database or other distributed data sources. The database interaction is done through data connection objects that also use HTTP as an exchange medium.

It is evident that the E-Health Technology Infrastructure is built on top of the Internet and its transfer protocols. The Internet is the backbone that glues and sustains

the communication between the different technical components of the system. Internet-founded architectures enjoy multiple benefits over proprietary infrastructures, including lower maintainability costs and higher system compatibility.

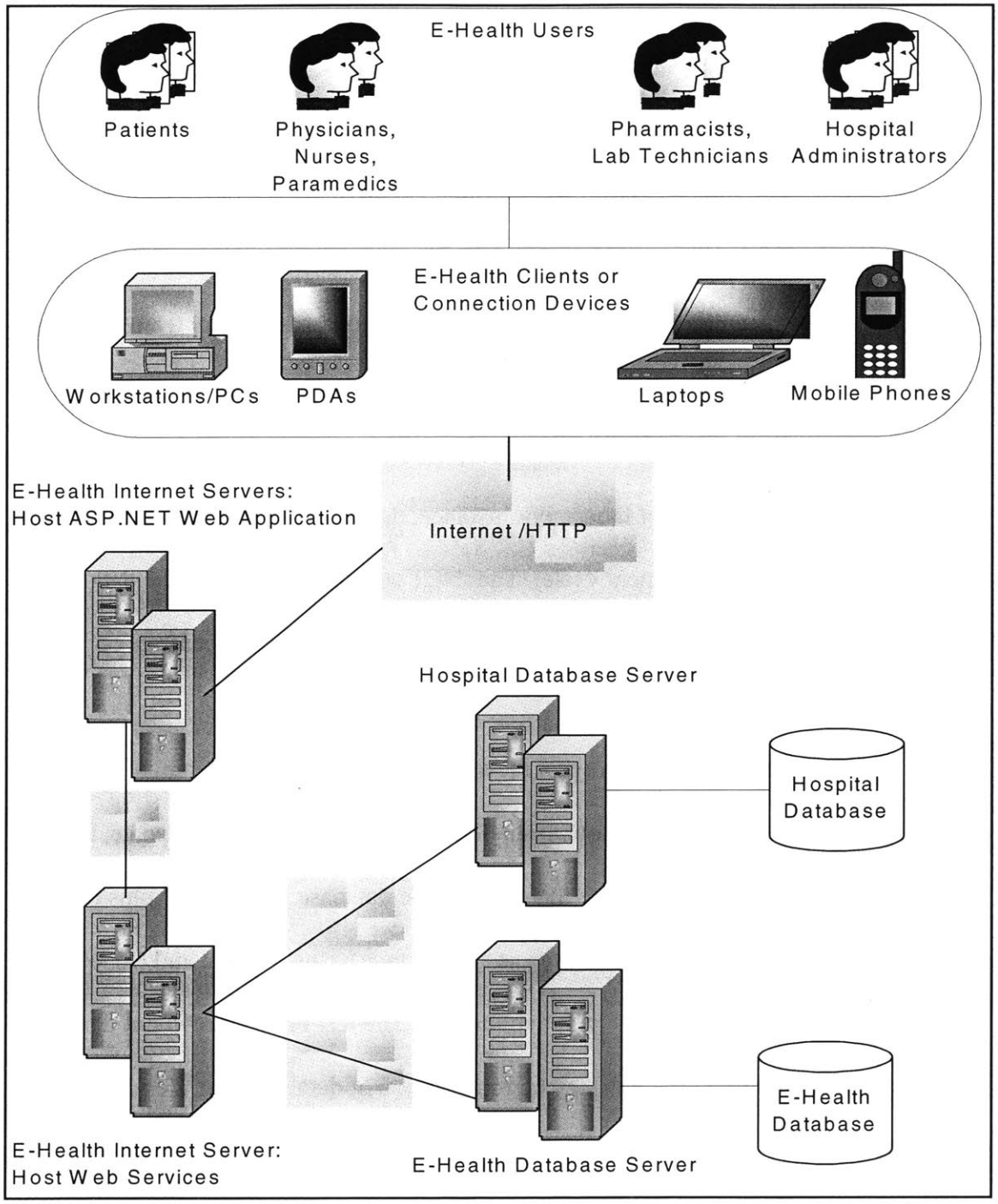

Figure 1. E-Health Technology Infrastructure

7 E-Health Development Cycle

The E-Health team decided to follow a spiral development model to address the complexity of the E-Health system. This model breaks the project into mini-projects, each concentrating on a different areas, and consequently reducing total risk. Time constraints only allowed the team to experience the first spiral iteration. During this iteration, the team followed an organized process where they determined objectives, evaluated alternatives and constraints, established deliverables, and developed the first E-Health prototype.

During the planning process, it was vital to identify that different team members have different skills, and a well-designed solution maximizes each member's skill set. Conveniently, the technologies selected for the project promote separation of presentation and data, facilitating division of labor. For instance, database modeling is a fundamental piece of the E-Health development cycle. The .NET technologies enable the data model to be separated from front-end developers, permitting members to concentrate on different development areas.

To create the E-Health prototype, the team had to design and create: (1) a database to store all the system information; (2) SQL queries for all add, update, and retrieve operations; (3) Web Services that will handle database interactions plus other functions; (4) Web application that serves as an interface to the system; and (5) Web Service references to link services with application. The rest of this document discusses these steps, concentrating on the design and integration of database and Web Service technologies.*

* The database and Web Service design presented in this document differs from the prototype presented to the project advisor. Modifications include revisions based on lessons from the first development cycle.

8 Database Modeling Process

8.1 Design Challenges

A database model for a complex information system needs to be comprehensive and flexible. This is especially true for health care institutions, where the subjectivity of the health care practitioner and the uniqueness of each patient's case are central aspects of the practice. Designing a database for such a system requires a deep understanding of the business processes and rules of the world being modeled. Market research helps understand needs, and map these into system's features. The E-Health team spent two to three months conducting personal interviews and researching the health care industry literature, before initializing the design phase.

Capturing processes and rules in a relational database model is certainly a challenge. A greater challenge is enclosing in a database the freedom of expression that is so essential for physicians. While examining a patient, most physicians write their observations and diagnosis in a blank sheet of paper, which gives them the freedom and flexibility to organize their thoughts in any possible way. Most physicians may look at some common points when examining a patient, but the information included in a patient's record is mostly subjectively selected. Physicians can include any and all the information they feel necessary, without constraints and without having to categorize the data.

Furthermore, every patient examination can follow very different paths according to the specific situation of given patient, the physician's knowledge, and the hospital's resources. When the workflow has so many possible paths, it is a challenge to design a database, as well as a Web application, in a way that will not constraint the physician or decision maker.

Finally, the health care industry changes rapidly. New drugs are introduced regularly to the market, while others are withdrawn. Innovative medical tests and equipments are made available, while other become obsolete. Biological discoveries change the way certain diseases should be evaluated and treated. A good database design for the E-Health system must guarantee database extensibility and the capacity to make frequent record changes. Moreover, these changes must not affect the current database structure, the relationship of the database with the application side, and most importantly, the data that is already stored.

8.2 Relational Database Modeling

In addition to having a detailed understanding of the world being modeled, a designer needs to understand database-modeling concepts. All information in a relational database model is represented as tables where each table is a set of records or rows. All records in a particular table have the same number of columns. Columns are also referred to as fields or attributes. A set of related tables forms a database. The following sections will discuss techniques and rules to create the E-Health database.

8.2.1 Entities and Attributes Selection

The first step in database modeling is identifying entities, in this case, the entities of the health care system. Entities can be thought of as system actors, objects, or other things with independent existence about which information will be stored in the system. A patient, a physician, a hospital visit, a prescription, and a laboratory test, are all examples of entities. Entities represent database tables. Each row in a table represents an occurrence of the entity, and each column represents a different attribute of the entity, such as the patient's name or the prescription's drug name.

It is good practice to come up with an entity and attribute naming convention. To identify entities, the E-Health database presents all entities with the first letter of each word capitalized and all words written together without spaces between them. Attributes will follow the same convention. For the purpose of this document, these will be enclosed by single quotation marks to be able to distinguish them from entities. For example, HospitalVisit is an entity and 'VisitId' is an attribute.

Each entity must have one attribute, or a combination of attributes, that uniquely identify each row in the table. This guarantees that it is possible to distinguish a particular record when performing different operations on the table. The unique attributes are known as primary keys. No primary key can have a null value, meaning that a value for this field must be provided for every row.

In the case of people-related entities, first and last names are usually not the best choice for a primary key. Many patients can have the same name. Instead, unique identifiers could be generated by the system incrementally, randomly, or according to some other factors. Identifiers can also include numbers that are unique by definition, such as the Social Security Number. In some cases, it is also beneficial to come up with an identifier convention. For example, when creating the E-Health prototype, some conventions were made. All patient ids start with the letter P, and are followed by a randomly generated 9-digit number. The same convention is true for the doctors, but instead, their ids start with the letter D.

8.2.2 Entity-Relationship and Diagrams

A central piece of relational database modeling is establishing relationships between entities. Foreign keys are an important concept in entity relationships. A foreign key is a column, or combination of columns, in a table that matches the primary

key of a related table. For example, if the 'PatientId' is a primary key in the Patient table, it will be a foreign key in any other table it is included.

There are three possible types of relationships: one-to-one, one-to-many, and many-to-many. One-to-one relationships mean that exactly one row corresponds to one row of the related table. For example, there could be one table that stores patients' names and another table that stores addresses and contact information. Both tables are referring to the same patient. The 'PatientId' in one table corresponds to exactly one 'PatientId' in the other table. It makes sense to merge these two tables, but in certain cases, it may be more efficient to separate them.

In one-to-many relationships, one row corresponds to many rows of the related table. For example, one patient can schedule many appointments, and each appointment corresponds to one patient. These are the ideal type relationships. Many-to-many relationships are not recommended because they force data repetition. In this type of relationship, many rows correspond to many rows of the related table. For example, a single medical procedure or test can be performed by many physicians, and the same physician can perform many other tests. If there is a test that includes three physicians, all test information will be repeated three times. It is better to create two one-to-many relationships so that each test-physician combination is unique. This is possible by creating an intermediary TestTester table. This example will be revisited in a later section.

Sketching the entities and their relationships is a fundamental part of the design process. It gives the designer and other developers a valuable visual tool. These sketches are called entity-relationship diagrams. The diagrams that will appear in subsequent

sections will identify primary and foreign keys with a PK and FK code, respectively. A connecting arrow between entities represents the relationships.

When establishing relationships, it is important to check the design with the rules of data normalization. These rules are a set of design standards that minimize data redundancy to prevent the introduction of inconsistent information. Inconsistencies can lead to data corruption and errors. In essence, the rules can be summarized as only including non-key columns that are facts of the primary key column of a table. For example, the Patient entity should contain an id that references their primary physician. It must not include any other physician-related information, such as his name or specialization. These non-key columns are not facts of the patient, but of the physician, and should only be included as columns of the Physician entity.

8.3 Data Manipulation and Retrieval

SQL, which stands for Structured Query Language, is a high-level language used for data manipulation. Through SQL commands, it is possible to retrieve and modify data from tables. The E-Health Web Services that are presented later in the document make extensive use of SQL commands and clauses. In fact, most E-Health Web methods are solely dedicated to perform data retrieval and manipulation with SQL.

Retrieval operations, normally referred to as queries, use the SQL SELECT statement to display the information requested. To specify the rows you want in a query, the WHERE clause allows identifying the criteria that a row must meet. Modification commands include INSERT, UPDATE, and DELETE. A key part of manipulating relational databases is the join operation. Joining two or more tables unites them as if

they were one during the duration of the operation by comparing the values in the specified columns. In this case, the WHERE clause is also used for value comparisons.

8.4 Advantages and Limitations of Relational Databases

The relational database model was chosen for the E-Health application for several reasons. Most members of the E-Health team had previous experience with the relational model and the SQL Server 2000. Due to time constraints, minimizing the learning curve was vital. For those members without experience, the relational model, with its row and column design, provided a simple, easy to learn user interface. In addition, the SQL Server 2000 has a well-known reputation for its good integration with Microsoft's development environments, including .NET. Although, .NET supports other database systems as easily as it does with the SQL Server, the team had limited time to complete the application with their existing knowledge.

Relational database technologies were designed to record business transactions accurately and reliably. Nevertheless, when it is necessary to manipulate massive quantities of data in real-time, a relational database may fall behind. The technology may not longer meet the needs of a business that require rapid data analysis. The essential problem with the relational model is that as the size of the database grows, query performance declines, cost of ownership increases, and it is impossible to maintain a real-time system. In a real-time system like E-Health, this is certainly a concern, especially when it is anticipated that the database will grow significantly every day. For instance, physicians perform several examinations each day, and each examination adds numerous rows to various database tables. Many patients will also enter several daily health measurements, adding hundreds or thousands of new records.

In addition to the problem of size, additional limitations are introduced when the database is complex. The relational model is easy to use, but only if it maps well to the application's data structures. If the structure is complex, mapping information to tables is like forcing a square box into a round hole. A great alternative is an object-oriented database model. The object-oriented model provides full-featured database programming capabilities combined with elements of object-oriented programming languages. The object's properties and methods encapsulate the complexity of the data. This model also has the object-oriented aspect of a class, which supports encapsulation, multiple inheritance, and abstract data types.

The object-oriented paradigm has the advantage of being a more natural way to model the real world. It provides more natural data structures, better maintainability, and reusability of code. They are also much more suitable for applications that have to process complex relationships. Object databases are extensible, which makes easy the definition and support of new data types. They also provide extended support for multimedia data types such as image, audio, and video, which will be necessary for future E-Health development cycles. For instance, these types allow storing x-ray and sonogram images, heart murmur audio files, and laparoscopic procedure videos.

Another alternative model is XML databases. These types of databases provide high performance with scalability. This model includes various levels of data complexity. XML databases are ideal solutions for applications that need to manage ever-changing data, such as E-Health. Database redesign is not necessary when introducing or modifying existing data structures. For applications that have embraced object-oriented programming and XML-based data transfer, these alternative models are

a natural choice. Future E-Health developments must carefully examine the migration to an object and/or XML database model.

9 E-Health Database Design

The E-Health prototype focuses on the fundamental interactions between the patients and the health care practitioners, specifically physicians. Thus, the database design that is presented in this document is primarily limited to this focus area. The discussion emphasizes the design of database tables including entities and attributes selection, and entity-relationship diagrams. To facilitate the database modeling process and discussion, the design was divided in groups. Each group represents a category of information that is needed to provide the proposed E-Health functionality.

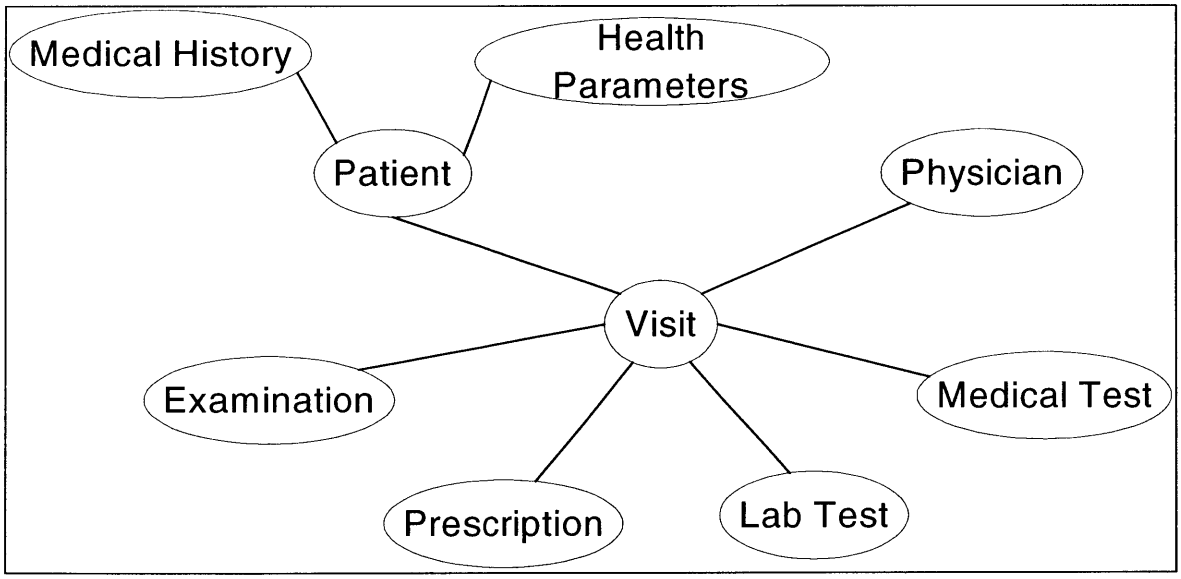

Figure 2. Database Design Overview

Figure 2 gives an overview of the main actors and groups of data of the E-Health system, and how they are related. The central link of the diagram is the Visit object, connecting all the pieces and actors together in a productive relationship. The following subsections present detailed discussions of the tables enclosed in each group in Figure 2.

9.1 Patients and Physicians

Every medical center needs to store and manage contact and demographic information about patients and physicians. This information is a central actor of the E-Health system. The interaction between the Patient and Physician entities drives most other database relationships. This section discusses these entities and their characteristics.

- **Patient** - The Patient entity is fundamental to the E-Health database because most other tables are directly related to this entity. Its attributes include 'PatientId', 'PhysicianId', 'FirstName', 'MiddleName', LastName', 'SSN', 'DateOfBirth', 'Age', 'Sex', 'Ethnicity', 'Weight', 'Height', 'MaritalStatus', and contact information such as 'Address', 'PhoneNumber' and 'Email' address. The primary key for this entity is a randomly assigned 'PatientId'. Though the Social Security Number (SSN) is a primary key candidate, this information is kept private. Finally, every patient has one primary physician, and, therefore, there is a field that stores the 'PhysicianId' of this physician. In addition, each physician can be assigned to many patients, creating a one-to-many relationship.
- **Physician -** The Physician entity is also central to the E-Health database because patients can receive the medical care they need through the physicians. The Physician's attributes include 'FirstName', 'MiddleName', LastName', 'Specialization', and contact information such as 'Address', 'PhoneNumber', 'Pager', and 'Email' address. The primary key of the table is 'PhysicianId'.

- **PhysicianVisitHour –** The system assumes that each physician has the same working hours every week. For example, every Wednesday, a particular physician examines patients from 8am to 10am, every 30 minutes. Thus, this entity stores 'PhysicianId', 'DayOfWeek', and 'Time'. The three attributes compose the table's primary key. 'DayOfWeek' represents one of seven possible days of the week. 'Time' indicates the starting time of each appointment. For the physician mentioned above, there would be four database records with the 'DayOfWeek' equal to 'Wednesday'. In this example, the 'Time' columns of these four records contain '8:00am', '8:30am', '9:00am', and '9:30am'. Every physician can have many records on the PhysicianVisitHour table, implying a one-to-many relationship.

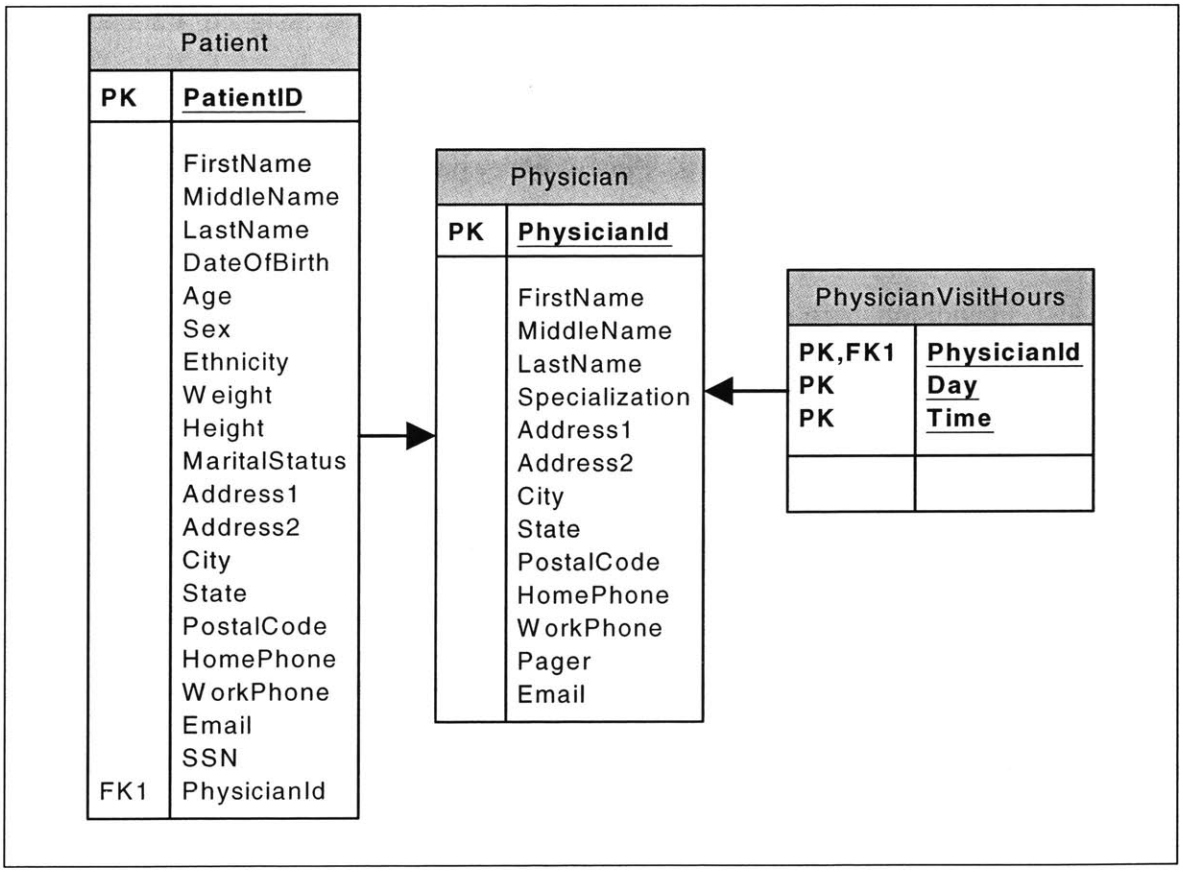

Figure 3. Patient-Physician Relationships

Figure 3 illustrates the relationship between patients and physicians, and their available visit hours.

9.2 Patient Medical History

Patients' medical history, including allergies and health conditions, is essential to better understand a patient's needs. For this reason, the database includes a group of tables to manage medical history.

9.2.1 Allergies

Allergies can be divided into two main groups: general allergies and drug-related allergies. Each group represents a database entity.

- **GeneralAllergy** - The attributes of the GeneralAllergy entity are 'PatientId', 'Allergy', 'Year', and 'Reaction. 'PatientId' refers to a patient in the Patient entity. 'Allergy' is the object that causes the allergic reaction. 'Year' specifies the year when the allergy was detected. 'Reaction' indicates the symptoms that the specified patient experiences when exposed to the allergy object. There is a one-to-many relationship between Patient and GeneralAllergy (i.e., any particular patient can have many allergies). Therefore, it is necessary to have a composite key to uniquely identify each record. The primary key combines 'PatientId' and 'Allergy'.
- **DrugAllergy** - The attributes of the DrugAllergy entity are 'PatientId', 'DrugId', 'Year', and 'Reaction'. 'DrugId' is a code that represents the ordered drug. Information about each drug is stored in the Drug entity. This entity is discussed in another section. The primary key is a combination of 'PatientId' and 'DrugId'. Both columns are also foreign keys that refer to the Patient and the Drug entity, respectively. A patient can have many drug

allergies, and many patients can be allergic to the same drug. Thus, there is a one-to-many relationship between Patient and DrugAllergy, and between DrugAllergy and Drug.

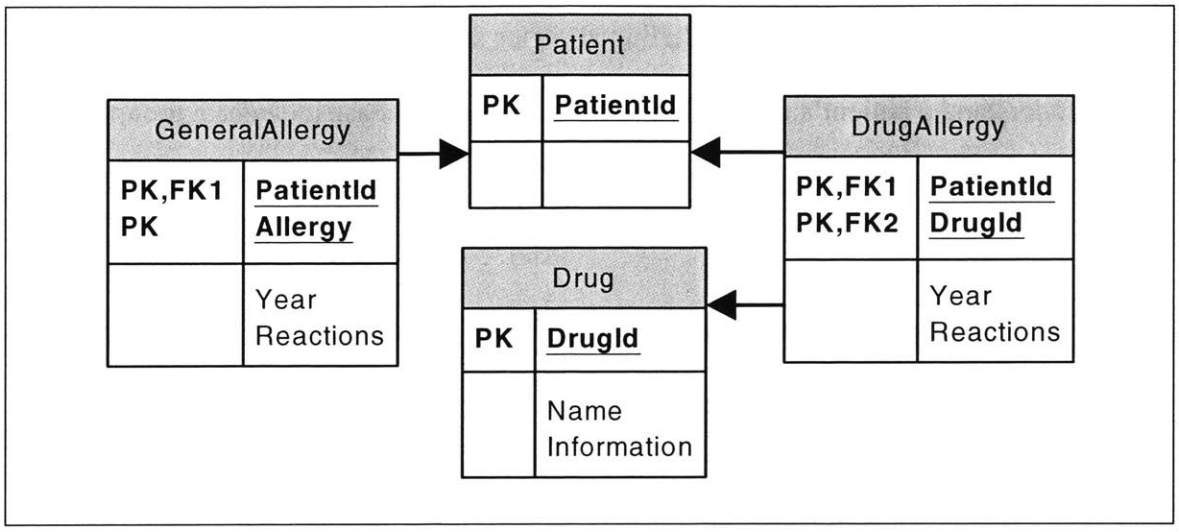

Figure 4. Patient-Allergy Relationships

Figure 4 above illustrates the discussed relationships. For simplicity, the Patient entity in the figure only includes the 'PatientId' field.

9.2.2 Health Conditions

Medical records store patients' health conditions and the disease history of their family members. The database includes two entities for this purpose. Figure 5 illustrates the relationships between the entities.

- **PatientHealthCondition** – The attributes of PatientHealthCondition are 'PatientId', 'ConditionId', and 'Comments'. 'ConditionId' is a code that represents the health condition that the patient has. The HealthCondition table holds information about each condition. This table is discussed shortly. 'Comments' stores additional information about a particular patient condition. The primary keys are 'PatientId' and 'ConditionId'. There is a one-to-many

relationship between this entity and the Patient entity, because a patient can have many conditions.

- **FamilyHealthCondition** - The FamilyHealthCondition entity contains the same attributes of the PatientHealthCondition table plus an additional field named 'FamilyMember'. This field specifies the patient's family member that is associated with the particular condition on record. The primary keys are 'PatientId', 'ConditionId', and 'FamilyMember'. A patient can have many family members with the same disease, and a particular family member can have multiple diseases.

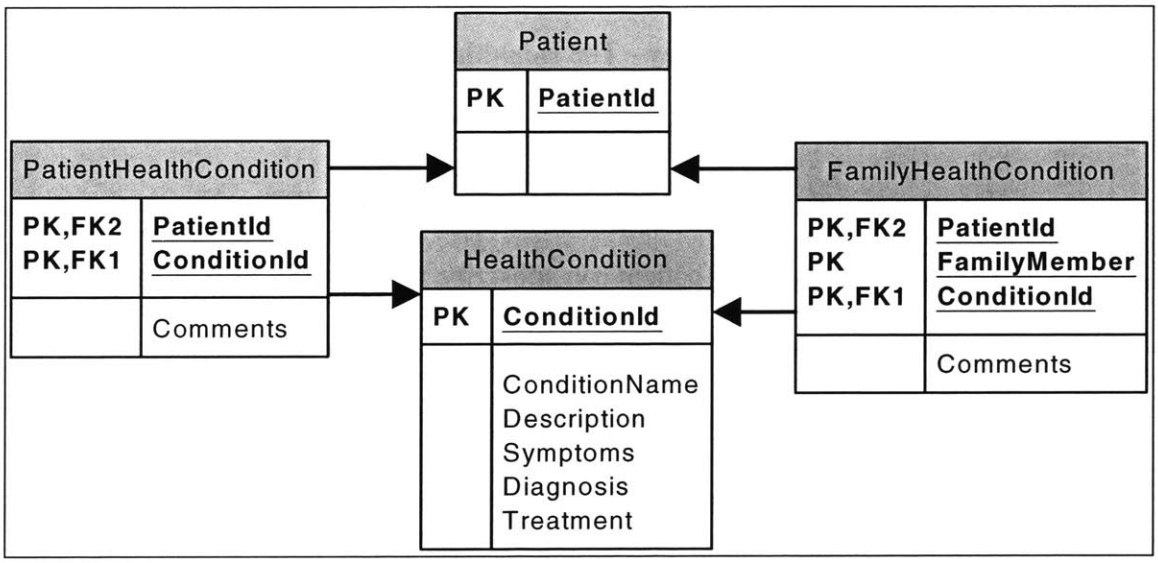

Figure 5. Patient-Health Conditions Relationships

- **HealthCondition** – The attributes of the HealthCondition entity are 'ConditionId', 'ConditionName', 'Description', 'Symptoms', 'Diagnosis', and 'Treatment'. The primary key is the 'ConditionId', which is a code that identifies each health condition on record. The other attributes are self-explanatory. The relationship between PatientHealthCondition and

HealthCondition is one-to-many. The same relationship is true for FamilyHealthCondition. Each record in HealthCondition can be linked to multiple records on PatientHealthCondition and on FamilyHealthCondition.

9.3 Health Insurance

Health insurance is a central but very complex piece of the health care system due to its many players and rules. In most cases, a patient must have health insurance to be eligible to receive medical care from a physician. As a benefit to hospitals, E-Health integrates the health insurance information into the database. The team took a simple approach to this matter, understanding that this area would need significant improvement in the future development phases.

- **HealthInsuranceCompany** - The HealthInsuranceCompany entity stores information about insurance companies. The attributes of this table are 'InsuranceCompanyId' 'CompanyName', 'Address', 'PhoneNumber', and 'ContactPerson'. Each patient can subscribe to the health plan of more than one company, and each company can have thousands of subscribers. This creates a many-to-many relationship between HealthInsuranceCompany and Patient, justifying the need for an intermediate table.
- **PatientInsurance** – PatientInsurance is an intermediate entity. It contains four attributes, which are 'InsuranceCompanyId', 'PatientId' 'InsurancePolicyNumber', and 'GroupPolicyNumber'. 'InsuranceCompanyId' is a foreign key that refers to a company in the HealthInsuranceCompany table. The policy numbers identify a patient's health plan in the respective insurance company. 'InsuranceCompanyId' and

'PatientId' represent the table's composite primary key. The PatientInsurance has a one-to-many relationship with HealthInsuranceCompany and with Patient. That is, each PatientInsurance record belongs to one particular patient and company.

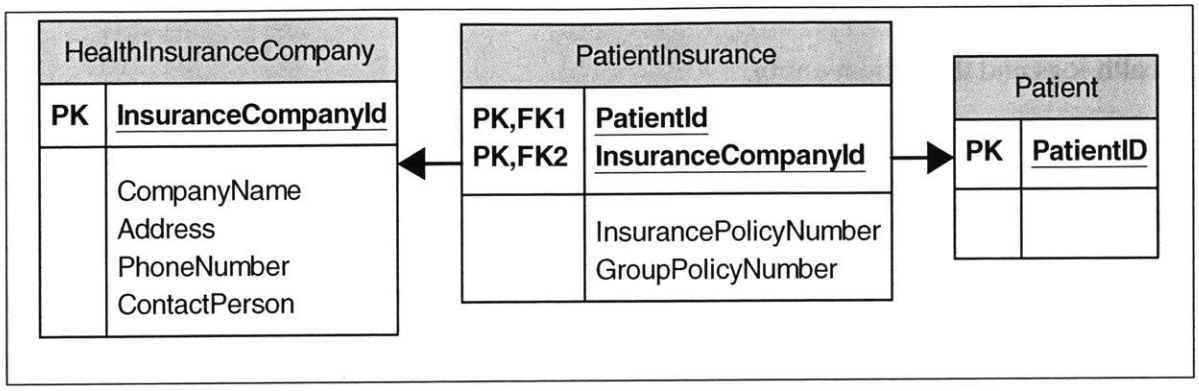

Figure 6. Patient-Health Insurance Relationships

Figure 6 illustrates the relationships between these tables. The relationship between a patient and an insurance company is much more complex than the one depicted in the figure. Insurance companies offer multiple plans with varied benefits and restrictions. These simple tables need further refinement in future phases. In addition, it would be necessary to model the interactions between insurance companies and health care institutions.

9.4 Vital Signs and Other Health Parameters

Taking regular health measurements and monitoring a patient's progress are vital elements to a complete health regimen. The E-Health system understands the importance of measurement management. Therefore, it provides storing and tracking functions for health parameters.

9.4.1 Health Parameters

The database contains seven entities corresponding to different health parameters. The idea behind these tables is to track a particular health parameter for a particular patient through time. All tables follow a similar logic and have a one-to-many relationship with the Patient entity. Figure 7 illustrates the relationships between the health logs and the Patient entity.

Each table has three common fields, which are 'PatientId', 'Date', and 'Comments' fields. 'Date' stores the date when the measurement was taken. 'Comments' provides flexibility to include any related information about a given measurement. In addition, each measurement has unique characteristics, and consequently, each table must have different attributes. For example, each entity must store the actual measurement value.

- **WeightLog** – The table contains the 'Weight' field. The primary keys are 'PatientId' and 'Date'. This allows patients to enter their weight once daily.
- **CaloriesLog –** The table contains the 'Calories' field. The primary keys are 'PatientId' and 'Date'. This allows patients to enter their daily caloric intake.

Some parameters are measured more than once daily and it is important to associate each measurement with a specific time. Thus, the following tables contain the 'Time' attribute. 'Time' together with 'PatientId' and 'Date' constitute the primary key of these tables to allow recording several measurements at different times during the same day.

- **GlucoseLog** – The table contains the 'GlucoseLevel' field.
- **BloodPressureLog** - The table contains the 'Systolic' and 'Diastolic' fields.
- **PulseLog** - The table contains the 'Pulse' field.

- **TemperatureLog** - The table contains the 'Temperature' field.

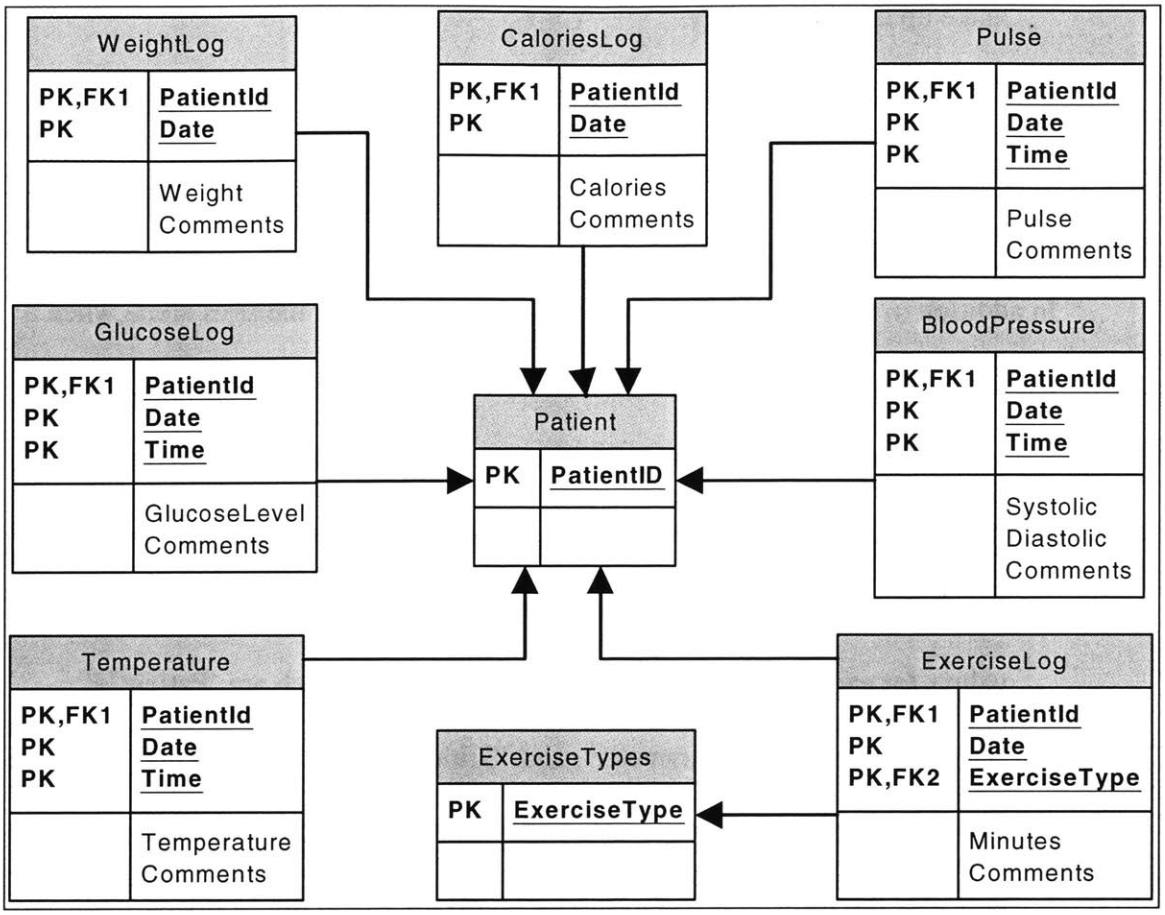

Figure 7. Patient-Health Logs Relationship

Finally, another table stores exercise activity. ExerciseLog is different from the other logs because there is no specific parameter to measure.

- **ExerciseLog** - Besides 'PatientId' and 'Date', ExerciseLog has two additional attributes, which are 'ExerciseType' and 'Minutes'. 'ExerciseType' specifies the type of exercise the patient performs. This field refers to a record in the ExerciseTypes table. 'Minutes' records the time in minutes that the patient spent on the given exercise activity. A patient can participate in more than

one type of activity per day. Thus, 'PatientId', 'Date', and 'ExerciseType' make up the primary key.

- **ExerciseTypes** – The table contains a varied list of exercises for patients to select. Its primary and only column is 'ExerciseType'.

9.4.2 Patient Alerts

In addition to tracking health parameters, the system also manages alerts when a given measurement exceeds an established critical value for a particular patient. Physicians are the ones that establish critical values. Different physicians may establish different values for the same patient based on their personal judgment.

- **ParameterAlertCriteria -** The ParameterAlertCriteria entity stores critical values for each patient. It contains four attributes, which are 'PatientId', 'PhysicianId', 'HealthParameter', and 'CriticalValue'. The 'PhysicianId' refers to the physician who created each alert on record. 'HealthParameter' indicates the parameter (e.g., glucose level) for which the alert was established. 'CriticalValue' is the value that the physician considers critical for a given patient. 'PatientId', 'PhysicianId', and 'HealthParameter' constitute the table's primary key. This combination of keys allows physicians to establish multiple alert criteria for multiple patients.
- **HealthParmeterAlert** - HealthParameterAlert is the entity that stores alerts when critical values are exceeded. Each alert is uniquely identified by an 'AlertId'. Other fields are 'PatientId', 'PhysicianId', 'HealthParameter', 'MeasurementValue', and 'Date'. 'MeasurementValue' stores the measurement entry that exceeded the established value. For example, if the glucose level for a patient is 130, and it exceeds its critical value, the

'MeasurementValue' field will store 130. 'Date' stores the date when the exceeding value was measured. Each record in HealthParameterAlert corresponds to exactly one established alert criteria, and each established alert could have many records on HealthParameterAlert. This implies a one-to-many relationship between the two tables.

Figure 8 illustrates the relationship between the patients, physicians, and alerts.

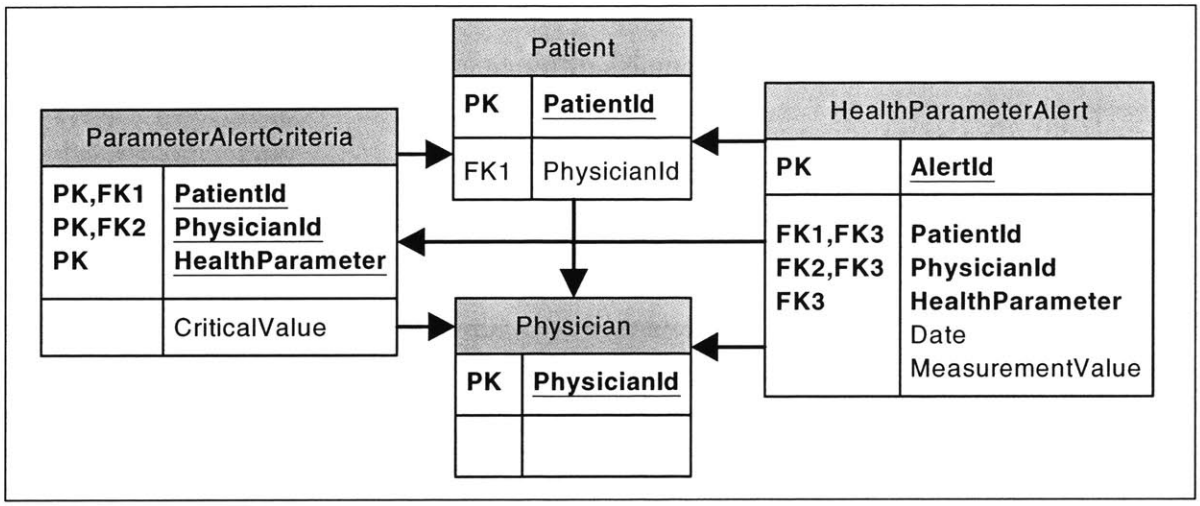

Figure 8. Patient-Alerts Relationships

The same underlying idea behind these alerts can be applied to laboratory or medical test results. For instance, if a blood sample measurement, such as hemoglobin, goes below a certain limit, the doctor can be immediately notified when the lab results are recorded. Currently, E-Health does not implement this functionality.

9.5 Hospital Visits and Examinations

A fundamental part of health care is encapsulated in a patient's visits to his or her physician. Due to the subjective nature of the health care practice, physicians need great flexibility during examinations to adapt to each case in the appropriate manner.

9.5.1 Hospital Visits

There are three principal types of visit, each having unique characteristics; therefore, the database contains three entities to model each type. These entities are HospitalizationVisit, EmergencyVisit, and ScheduledVisit. The three tables contain 'PatientId', 'VisitId', and 'Comments'. 'VisitId' stores a randomly generated id that uniquely identifies each visit on record. 'Comments' allows storage of additional information related to the particular visit, including visit purpose. In addition, each visit has unique characteristics, and thus contains different columns. Figure 9 illustrates the visit entities and its relationships.

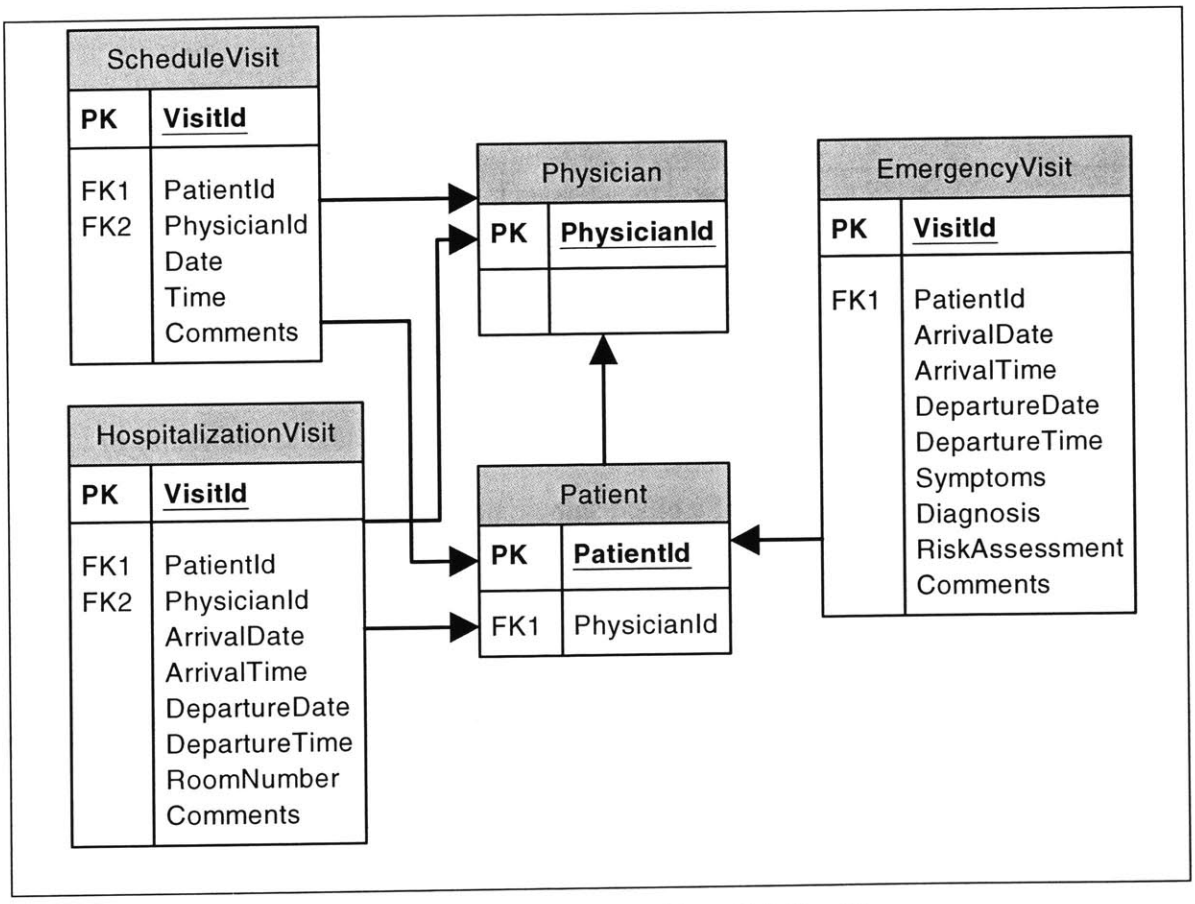

Figure 9. Health Parameters-Alerts Relationships

- **ScheduledVisit** - Contains the visit 'Date' and 'Time', and the 'PhysicianId' of the physician being visited. E-Health Web Services and application focuses on ScheduledVisit.

The HospitalizationVisit and EmergencyVisit entities have other common fields. These are patient's 'ArrivalDate', 'ArrivalTime', 'DepartureDate' and 'DepartureTime' to the hospital or emergency room.

- **HospitalizationVisit** – Contains the 'RoomNumber' where the patient is staying at the hospital, and the 'PhysicianId' of the physician in charge of the hospitalization.
- **EmergencyVisits** – In emergency visits, there is a need to include information about the patient's 'Symptoms', preliminary 'Diagnosis', and 'RiskAssessment'.

9.5.2 Examinations

During a hospital visit, a patient can go through several types of examination, each completed by a different caregiver.

- **Examination** - The Examination entity provides the flexibility to store information about different types of examinations. The table's attributes are 'ExaminationNumber', 'VisitId', 'Date', 'ExaminerId', 'ExaminationTypeId', 'Observations', and 'Comments'. The 'ExaminationNumber' field starts with the value '1' when the physician orders the first prescription for the specified visit. This value increases sequentially with every additional prescription ordered in the given visit. 'VisitId' links each examination to the visit when it took place. 'Date' stores the date when the examination was completed. During a scheduled visit, the examination date will be the same as the visit

date. In the case of hospitalization and emergency visits, the date is between the arrival and departure dates. 'ExaminerId' represents the id of the physician, nurse, or paramedic that completed the examination. 'ExaminationTypeId' specifies the type of examination that was completed. Information about each examination type is stored in the ExaminationType entity. 'Observations' stores important observations made during the examination. For example, if a doctor examines a patient's eyes, he can write about the condition of the eyes in that field. The 'Comments' field stores additional remarks made by the practitioner. Each record is uniquely identified by 'ExaminationNumber' and 'VisitId'. A record in the Examination table corresponds to exactly one visit, but each visit can have multiple examinations. This creates a one-to-many relationship between ScheduledVisit and Examination.

- **ExaminationType** - ExaminationType contains a list of all possible examinations. It contains 'ExaminationTypeId', which is the primary key, 'ExaminationName', and 'ProcedureDescription'.

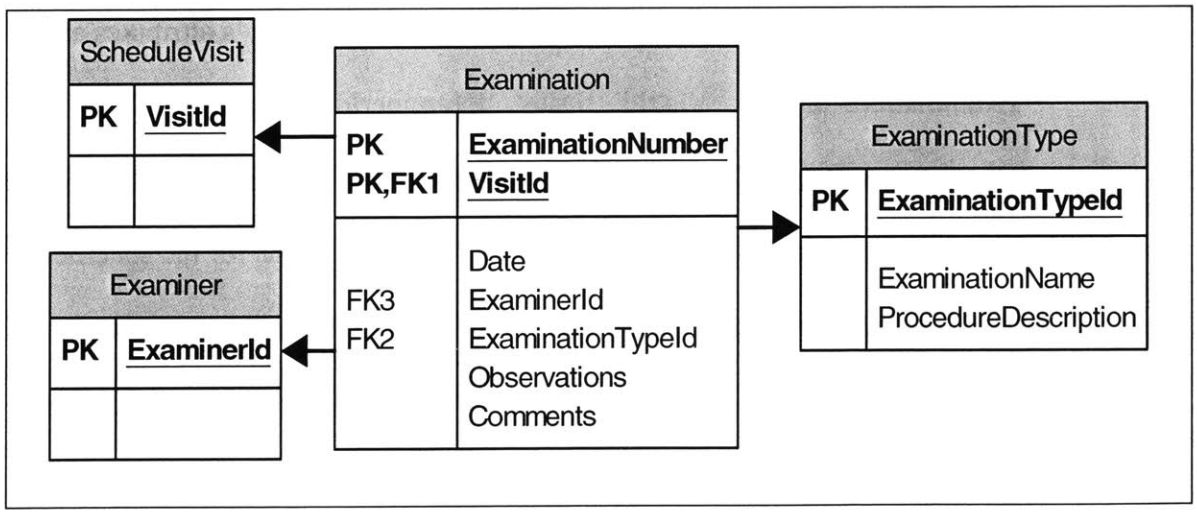

Figure 10. Visit-Examination Relationships

Figure 10 illustrates the relationships of the Examination table. For simplicity purposes, the Examiner table encloses physicians, nurses, and paramedics.

9.6 Patient Prescription Orders

A fundamental part of any medical visit is the prescriptions ordered by the physician. The E-Health database contains two tables that manage prescription-related information.

- **Prescription Orders -** The PrescriptionsOrder entity stores the necessary information to complete the prescription ordering process. A physician can order several prescriptions for a patient in a given visit, creating a one-to-many relationship between PrescriptionOrder and any of the visit entities. The table's attributes are 'PrescriptionNumber', 'VisitId', 'OrderDate', 'PhysicianId', 'DrugId', 'DrugQuantity', 'Dose', 'Refills', 'PrescriptionPurpose', and 'Special Instructions'. 'Prescription Number' is a unique identifier for each prescription ordered during a given visit. The 'PrescriptionNumber' field starts with the value '1' when the physician orders the first prescription for the specified visit. This value increases sequentially with every additional prescription ordered in the given visit. The 'VisitId' is a foreign key referring to the visit where the prescription was ordered. 'OrderDate' stores the date the order was submitted by the physician. 'PhysicianId' refers to the physician that submitted the order. 'DrugId' represents the medication being prescribed. This field refers to a record of the Drug entity. 'DrugQuantity' is the total amount of tablets, capsules, or liquid that must be provided to the patient. 'Dose' is the drug amount the patient

should take at each specified time interval. 'Refills' indicates how many times the patient is allowed to reorder the prescription. 'PrescriptionPurpose' allows the physician to specify the purpose of the prescribed drug. 'SpecialInstructions' provides the flexibility to add additional specifications or comments about the medication usage. The primary key of the table is composed of 'VisitId' and 'PrescriptionNumber'.

- **Drug -** The Drug table stores 'DrugId', 'DrugName', 'ChemicalName', 'CommonUses', 'Directions', 'Cautions', 'SideEffects', and 'Additional Information'. 'DrugId' is the primary key. The other fields are used to describe the drug purpose and its usage. There is a one-to-many relationship between the PrescriptionOrder and Drug.
- **Patient Prescriptions -** The PatientPrescription entity stores further information about a particular order and its subsequent steps. A record in the PrescriptionOrder entity can have more than one related records on the PatientPrescription table, creating a one-to-many relationship. This is because a particular prescription order can be repeated several times. The repetition limit is imposed by the value in the 'Refills' field of the PrescriptionOrder table. The table's attributes are 'VisitId', 'PrescriptionNumber', 'RefillNumber', 'RequestDate', 'CompletedDate', 'PickupDate', 'PharmacyId', 'PharmacistId', and 'Comments'. The first two attributes identify a particular record in PrescriptionOrder. 'RefillNumber' identifies the number of times a particular order has been refilled. This field starts with the value '0' when a prescription is completed for the first time. The value

increases sequentially with each refill for the given prescription. 'RequestDate', stores the date that the patient requested the medication from the pharmacy. 'CompletedDate' indicates the date the order was completed by the pharmacy. 'PickupDate' is the date the patient picked up the medication. 'PharmacyId' refers to the pharmacy where the medication was ordered. 'PharmacistId' links the order to the particular pharmacist that completed it. 'Comments' allows the pharmacist to include any necessary commentaries about the order. The primary keys of PatientPrescription are 'VisitId', 'PrescriptionNumber', and 'RefillNumber'.

Figure 11 illustrates relationships between the different prescription tables.

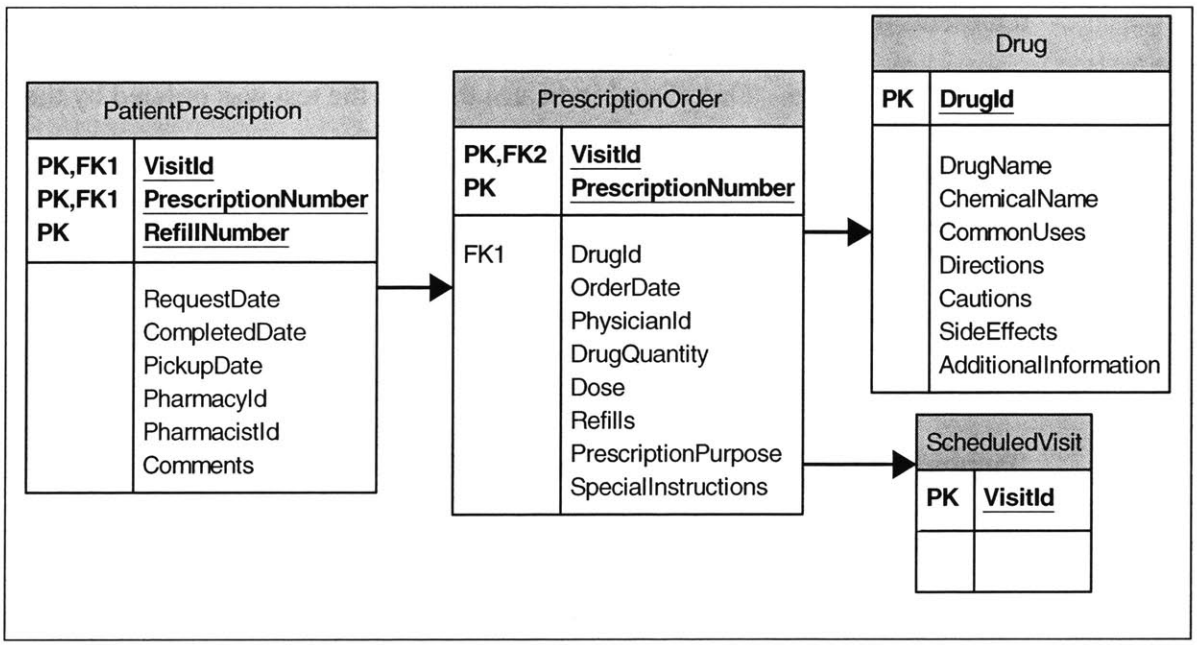

Figure 11. Prescription Orders Relationships

9.7 Laboratory Tests

Laboratory test are important to help diagnose and adequately treat medical conditions. The E-Health database stores lab test orders submitted by physicians, and the respective test results. Physicians can order multiple laboratory tests per visit. It is even

possible to order the same lab test more than once during a lengthy visit, such as a hospitalization. This creates a one-to-many relationship.

- **LabTestOrders** - The LabTestOrder entity stores the necessary information to complete the test ordering process. The table's attributes are 'VisitId', 'LabTestNumber', 'LabTestId', 'OrderDate', 'PhysicianId', and 'Comments'. The 'VisitId' refers to the specific visit where the test was ordered. 'LabTestNumber' is a unique identifier for each test ordered during a given visit. The 'LabTestNumber' field starts with the value '1' when the physician orders the first test for the specified visit. This value increases sequentially with every additional lab test ordered in the given visit. 'LabTestId' represents the lab test being ordered. This field refers to a record in the LabTestType table. 'OrderDate' indicates the date the test was ordered by the physician indicated in the 'PhysicianId' field. 'Comments' holds special instructions or commentaries about the specific test. The primary keys of the table are 'VisitId' and 'LabTestNumber'.
- **LabTestType** – The table contains 'LabTestId', 'TestName', 'Description', 'Purpose', 'Procedure', and 'Preparation'. 'LabTestId' is the primary key. The other fields are used to describe the test. There is a one-to-many relationship between the LabTestOrder and LabTestType.

The database also provides tables that store lab test results. Given that each lab test has unique attributes, there are different tables for each test type. For example, the GlucoseToleranceTest table stores glucose levels for regular time intervals. The CellBloodCountTest table stores the values of different cell types at a given time. There

are, however, six common fields among all test types' tables. The common fields are 'VisitId', 'LabTestNumber', 'TestDate', 'LaboratoryId' 'LabTechnicianId', and 'Comments'. 'VisitId' and 'LabTestNumber' constitute the primary keys of all test result tables. 'TestDate' indicates the date the test was completed. 'LabTechnicianId' refers to the technician that evaluated the test sample. 'LaboratoryId' refers to the lab facility where the test was evaluated. 'Comments' provides a space to add additional observations.

Each record on LabTestOrder corresponds to exactly one record on one of the test results tables, establishing a one-to-one relationship among the tables. This is one of the rare cases where this type of relationships is used. It is necessary to have a one-to-one relationship because each test follows a different format, and cannot be grouped into one uniform table. Figure 12 illustrates relationships between patients' visits, lab test orders, and their results.

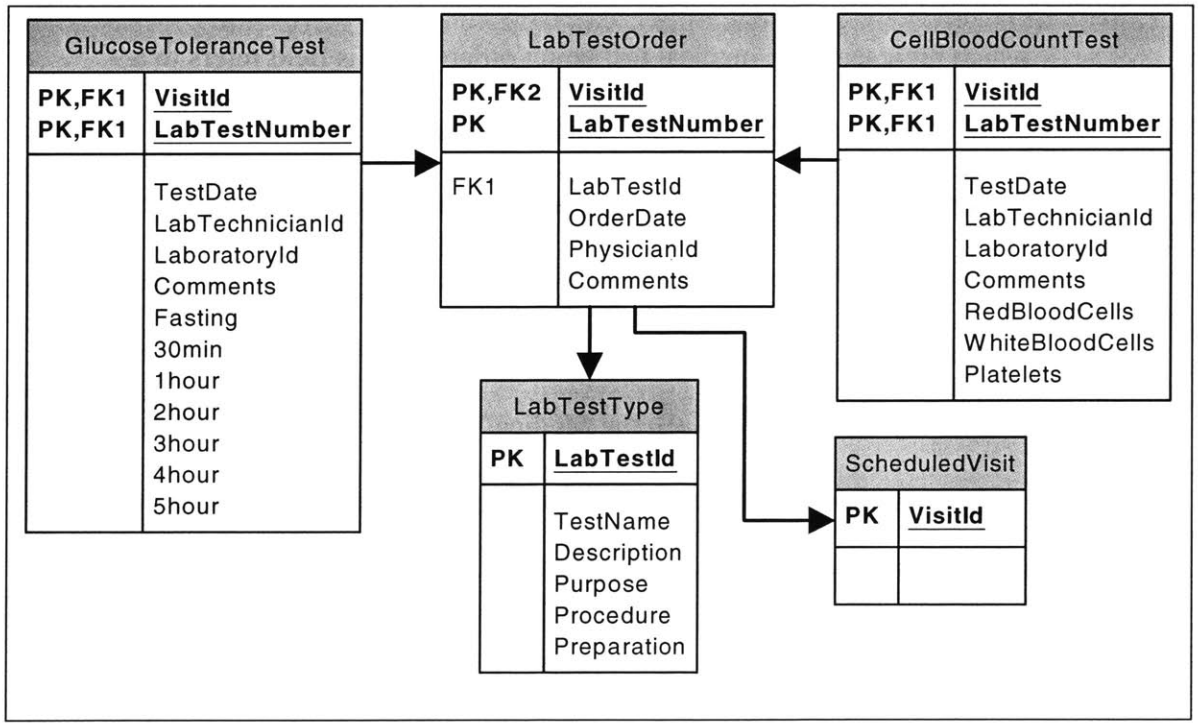

Figure 12. Laboratory Tests Relationships

9.8 Medical Tests

Medical tests include simple and sophisticated procedures that aid in the diagnosis of a condition or disease and help determine its severity. Test examples include sonograms, electrocardiograms, stress tests, and x-rays.

- **MedicalTestOrder** - The MedicalTestOrder table resembles the LabTestOrder table. It has a dual primary key composed of the 'VisitId' and a 'TestNumber'. The 'VisitId' refers to the specific visit where the test was ordered, and the 'TestNumber' is a unique identifier for each test ordered during a given visit. The 'TestNumber' field starts with the value '1' when the physician orders the first test for the specified visit. This value increases sequentially with every additional test order submitted in the given visit. Like in the LabTestOrder table, the MedicalTestOrder table also stores 'OrderDate', 'PhysicianId', and 'Comments'. It also stores 'MedicalTestId', which represents the medical test being ordered. This field refers to a particular record in the MedicalTestType table.
- **MedicalTestType** – This table contains 'MedicalTestId', 'TestName', 'Description', 'Purpose', 'Procedure', and 'Preparation'. 'MedicalTestId' is the primary key. The other fields are used to describe the test. There is a one-to-many relationship between the MedicalTestOrder and MedicalTestType.

The database provides tables that store medical test results. Given that each medical test has unique attributes, several tables store different portions of information about each test.

- **TestObservationsResult** – The TestObservationsResults stores test observations and results in the form of text. Each record on MedicalTestOrder corresponds to exactly one record on this table, establishing a one-to-one relationship. The fields are 'VisitId', 'MedicalTestNumber', 'TestDate', 'Observations', and 'Results'. The primary keys of the table are 'VisitId' and 'MedicalTestNumber'. 'TestDate' indicates the date the test was completed.
- **TestImage** - The TestImage table provides the flexibility to store images for different tests regardless of their type. For example, it can store x-rays and sonograms images. The table stores 'VisitId', 'MedicalTestNumber', 'ImageNumber', 'ImageFileName', and 'Obserrvations'. The first two fields link the record with a specific test order. 'ImageNumber' field starts with the value '1' when the first image is stored. The value increases sequentially with every additional image. 'ImageFileName' stores the name of the file that contains the test image. 'Observations' allows any comments about the particular image. The primary key is constructed of 'VisitId', 'MedicalTestNumber', and 'ImageNumber'. Many images can be created in a particular test, thus each record on MedicalTestOrder can have many associated records on TestImage. This establishes a one-to-many relationship between the tables.
- **TestTester** - The TestTester table stores the physicians and technicians that were involved in a given test. Contrary to a lab test where a single technician evaluates a blood sample, a medical test can be administered by several

caregivers. The table fields are 'VisitId', 'MedicalTestNumber', and 'TesterId'. The first two fields identify a particular test order. 'TesterId' represents the technician or physician involved in the procedure. There is a one-to-many relationship between TestTester and the MedicalTestOrder and Tester entity. For simplicity purposes, the Tester entity is used in this case to represent physicians, nurses, and technicians.

Figure 13 illustrates relationships between patients' visits, medical test orders, and their results.

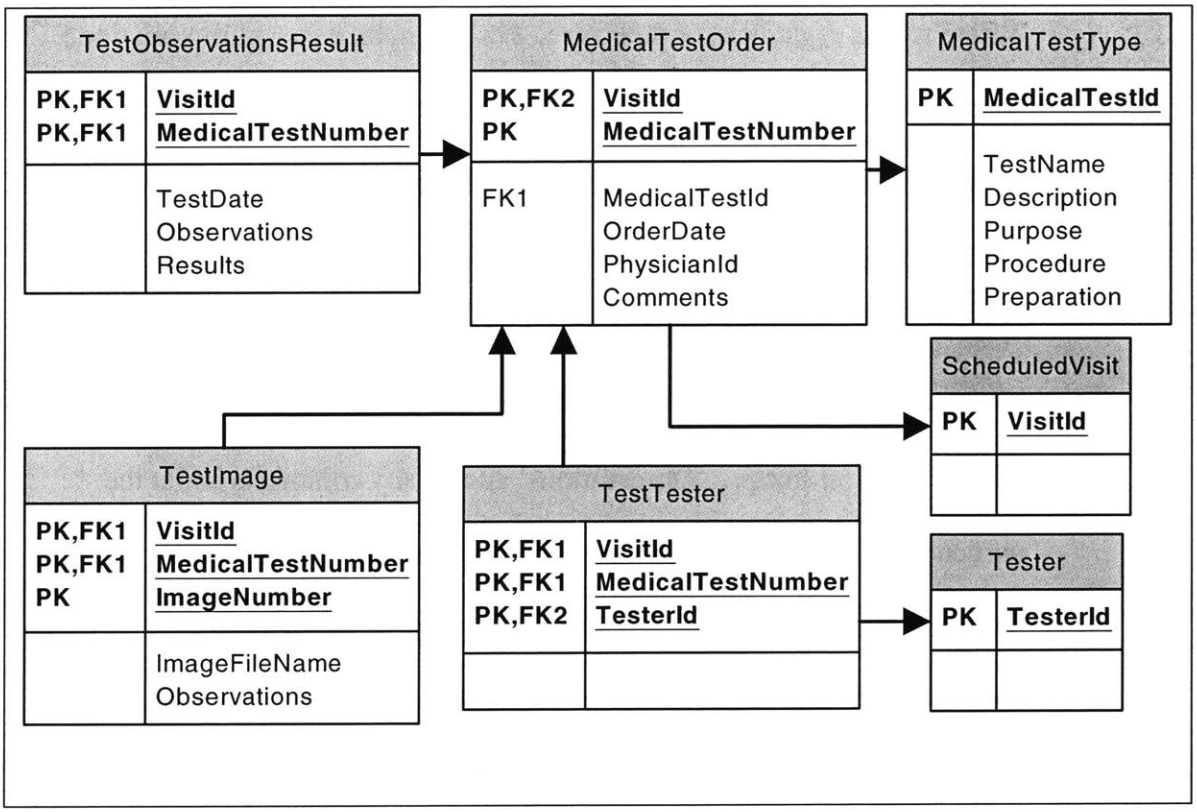

Figure 13. Medical Test Relationships

10 E-Health Web Services Design and SQL Queries

Web Services are a vital piece of the E-Health solution. These are designed to perform different remote computing functions, particularly database manipulations. Each Web method is associated with a particular SQL query. Some methods contain simple

single-table queries. Other methods perform complex operations that include multiple queries involving joins, DataSet comparison, and internal method calls. The E-Health Web methods have two return types. Methods that retrieve data have a DataSet return type. Methods that include data insertion return a Boolean indicating the success or failure of the operation.

There is a design tradeoff between the generality and specificity of the Web methods. Most methods were designed to be used optimally with the E-Health Web and mobile applications because that was the team's ultimate goal. Although the presentation layer is separated from data manipulations, the Web Service designers thought about how E-Health end users want to view and organize their information. Thus, some methods were designed specifically for the E-Health interfaces. Nevertheless, most methods are also general enough to be used with other applications. Some examples of this design tradeoff are discussed later.

E-Health has seven Web Services: PatientInfo, HealthTools, Scheduling, Examination, Prescription, Laboratory, and MedicalTest. The database served as a model to structure the different services. Therefore, each service manages the different database groups that have been discussed. Each service and its methods are discussed in the following sections.

10.1 PatientInfo Service

The methods in the PatientInfo Web Service manage patients' personal and medical information. All methods enclose a simple SQL query that retrieves records from the different tables related to the Patient entity. The basic functionality of each method is explained below.

- **PatientInformation** – Takes a 'PatientId' and retrieves a DataSet from the Patient table that contains one record with information about the specified patient.
- **PatientInsurancePlans** - Takes a 'PatientId' and retrieves a DataSet from the PatientInsurance table that contains one or more records with health insurance information of the specified patient.
- **PatientPrimaryPhysician** - Takes a 'PatientId' and retrieves a DataSet from the Physician table that contains one records with information about the primary physician of the specified patient. First, the method searches in the Patient table for the 'PhysicianId', and then uses it to retrieve the record from the Physician table.
- **PhysicianPatients** - Takes a 'PhysicianId' and retrieves a DataSet from the Patient table that contains zero or more records with information about the patients that have as primary physician the one with the specified ID.
- **PatientGeneralAllergies** - Takes a 'PatientId' and retrieves a DataSet from the GeneralAllergy table that contains zero or more records about the general allergies of the specified patient.
- **PatientDrugAllergies** - Takes a 'PatientId' and retrieves a DataSet from the DrugAllergy table that contains zero or more records about the drug allergies of the specified patient.
- **PatientHealthConditions** - Takes a 'PatientId' and retrieves a DataSet from the PatientHealthCondition table that contains zero or more records about the health conditions of the specified patient.

- **PatientFamilyHealthConditions -** Takes a 'PatientId' and retrieves a DataSet from the FamilyHealthCondition table that contains zero or more records about the family health conditions of the specified patient.

10.2 HealthTools Service

There are fourteen web methods in the HealthTools Web Service that deals with one of the seven health parameter tables. Each table has one corresponding method to view existing records and another to add new records. In addition, there are four methods to manage parameter alerts.

The methods that add new records have similar implementation logic. After inserting the new measurement, the method calls a private method to search for the patient's alerts that are in the ParameterAlert Criteria table. A single measurement can exceed the established criteria of several physicians. The private method returns a DataSet containing the records of any alert criteria that was exceeded.

The update method verifies if the DataSet is empty or not. If it is not empty, it calls another private method to insert new records in the Health ParameterAlert table corresponding to each of the exceeding alerts. The private method generates a random 'AlertId' for each record to be the primary key of the HealthParameterAlert. A description for each method follows.

- **ViewBloodPressureLog** - Takes a 'PatientId' and retrieves a DataSet from the BloodPressureLog table that contains zero or more records with all blood pressure entries of the specified patient.
- **UpdateBloodPressureLo**g - Takes 'PatientId', 'Date', 'Time', 'BloodPressure', and 'Comments' to input a new blood pressure entry. A

SQL INSERT statement creates a new record that stores the necessary fields in the BloodPressureLog. The method returns a Boolean type that indicates whether the insertion was successful or not.

- **ViewPulseLog** - Takes a 'PatientId' and retrieves a DataSet from the PulseLog table that contains zero or more records with all pulse entries of the specified patient.
- **UpdatePulseLog** - Takes 'PatientId', 'Date', 'Time', 'Pulse', and 'Comments' to input a new pulse entry. A SQL INSERT statement creates a new record that stores the necessary fields in the PulseLog. The method returns a Boolean type that indicates whether the insertion was successful or not.
- **ViewWeightLog** - Takes a 'PatientId' and retrieves a DataSet from the WeightLog table that contains zero or more records with all weight entries of the specified patient.
- **UpdateWeightLog** - Takes 'PatientId', 'Date', 'Weight', and 'Comments' to input a new weight entry. A SQL INSERT statement creates a new record that stores the necessary fields in the WeightLog. The method uses the new weight entry to update the Patient table through another SQL query. This table stores the most recent weight for a patient. The method returns a Boolean type that indicates whether the insertion was successful or not.
- **ViewGlucoseLog** - Takes a 'PatientId' and retrieves a DataSet from the GlucoseLog table that contains zero or more records with all glucose level entries of the specified patient.

- **UpdateGlucoseLog -** Takes 'PatientId', 'Date', 'Time', 'GlucoseLevel', and 'Comments' to input a new glucose entry. A SQL INSERT statement creates a new record that stores the necessary fields in the GlucoseLog. The method returns a Boolean type that indicates whether the insertion was successful or not.
- **ViewCaloriesLog** - Takes a 'PatientId' and retrieves a DataSet from the CaloriesLog table that contains zero or more records with daily calories amounts of the specified patient.
- **UpdateCaloriesLog -** Takes 'PatientId', 'Date', 'Calories', and 'Comments' to input a new calorie entry. A SQL INSERT statement creates a new record that stores the necessary fields in the CaloriesLog. The method returns a Boolean type that indicates whether the insertion was successful or not.
- **ViewTemperatureLog** - Takes a 'PatientId' and retrieves a DataSet from the TemperatureLog table that contains zero or more records with all temperature entries of the specified patient.
- **UpdateTemperatureLog -** Takes 'PatientId', 'Date', 'Time', 'Temperature', and 'Comments' to input a new temperature entry. A SQL INSERT statement creates a new record that stores the necessary fields in the TemperatureLog. The method returns a Boolean type that indicates whether the insertion was successful or not.
- **ViewExerciseLog** - Takes a 'PatientId' and retrieves a DataSet from the ExerciseLog table that contains zero or more records with all reported exercise activity of the specified patient.

- **UpdateExerciseLog** - Takes 'PatientId', 'Date', 'ExerciseType', 'Minutes', and 'Comments' to input a new exercise entry. A SQL INSERT statement creates a new record that stores the necessary fields in the ExerciseLog. The method returns a Boolean type that indicates whether the insertion was successful or not.
- **EstablishPatientAlertCriteria** – Takes 'PatientId', 'PhysicianId', 'HealthParameter', and 'CriticalValue'. A SQL INSERT statement creates a record that stores these fields in the ParameterAlertCriteria table. The method returns a Boolean type that indicates whether the insertion was successful or not.
- **DeletePatientAlertCriteria –** Takes 'PatientId', 'PhysicianId', and 'HealthParameter'. A SQL DELETE statement eliminates the record these fields refer to. The method returns a Boolean type that indicates if the operation was successful or not.
- **ViewPatientAlerts** – Takes a 'PatientId' and retrieves a DataSet from the HealthParmeterAlert table that contains zero or more records with all reported alerts for the specified patient.
- **ViewPhysicianPatientAlerts -** Takes a 'PhysicianId' and retrieves a DataSet from the HealthParmeterAlert table that contains zero or more records with all reported alerts that were established by the specified physician.

10.3 Scheduling Service

The Scheduling service allows patients and physicians to arrange and manage their medical appointments. The methods entail a series of table joins, comparisons, and additional computation in addition to database manipulation.

- **PhysicianAvailableTimes** – Takes 'PhysicianId' and 'Date' and returns a DataSet that contains zero or more records with the available times of the physician on the specified date. The data returned is not retrieved directly from an existing database table. First, a query retrieves the physician's working hours during the specified date. This data is found in the PhysicianVisitHours table. Then, another query retrieves the times when the physician has scheduled appointments for the specified date. This data is found in the ScheduledVisit table where all scheduled visits are recorded. The web method compares the times returned by both queries, and identifies the times when the physician is still available. A DataSet is created to store these times, and then it is returned to the calling function.
- **CreateAppointment**– Takes in 'PatientId', 'Date', 'Time', and 'Comment' to schedule a new appointment. The E-Health prototype only allows making appointments with primary physicians. The 'PhysicianId' is not needed as an input to this method. It identifies who is the patient's primary physician by calling the PatientPrimaryPhysician method of the PatientInfo Web Service. The method also checks if the patient is available at the specified date and time. This is done by comparing the given date and time with that of currently scheduled appointments, which is stored in the ScheduledVisit table. If the patient is available, a random 'VisitId' is generated to uniquely identify the

record. A private method named VisitIdGenerator creates this identification. This method verifies that that the generated id does not exist in the database. Finally, a SQL INSERT statement creates a new record that stores the necessary fields in the ScheduledVisit table. The method returns a Boolean type that indicates whether the insertion was successful or not.

- **PhysicianAppointments** - Takes 'PhysicianId' and 'Date' to retrieve a DataSet from the ScheduledVisit table that contains zero or more records with information about the physician's scheduled appointments for the specified date.
- **PatientFutureAppointments** – Takes a 'PatientId' and retrieves a DataSet from the ScheduledVisits table that contains zero or more records with information about all currently scheduled appointments for the given patient. The 'Date' column of the selected records contains the current date or a future date.
- **PatientPastAppointments** – Takes a 'PatientId' and retrieves a DataSet from the ScheduledVisit table that contains zero or more records with information about all past appointments for the given patient. The 'Date' column of the selected records contains the current date or a past date.

It is interesting to notice the difference in design between the three methods that return appointments. Physicians have many appointments every day, and the best way to organize these is by date. Thus, there is a method to retrieve appointments by date. On the contrary, patients may just have one or two scheduled appointments, and it is more

efficient to show them a complete list of all future appointments. The purpose of the PatientPastAppointments method is shown later.

This difference in design shows the tradeoff between the generality and specificity of the methods. Users' needs play a significant role in deciding where to establish the tradeoff. The three methods interact with the same table, but are designed for different functions. If a general method were used for all three functions, it would imply further SQL manipulations in the ASP.NET application. The Web Services were designed so that E-Health ASP.NET developers do not have to worry about data manipulations.

10.4 Examination Service

The Examination service contains a series of methods to record and view data related to patients' examinations.

- **ExaminationTypes** – Takes no argument and returns a DataSet that contains a list of all examination types stored in the ExaminationType table.
- **CreateExamination** – Takes 'VisitId', 'ExaminationTypeId', 'ExaminerId', 'Date', 'Observations', and 'Comments'. The method calls another a private non-web method named ExaminationNumberGenerator to create a unique 'ExaminationNumber'. First, the private method verifies if there are other examinations linked to the specified visit. Then, it generates the 'ExaminationNumber' based on the number of previous examinations. For example, if there were two previous examinations in the given visit, the number of the new record will be '3'. A SQL INSERT statement creates a

record that stores these fields in the Examination table. The method returns a Boolean type that indicates whether the insertion was successful or not.

- **ViewVisitExaminations** – Takes a 'VisitId' and returns a DataSet from the Examination table that contains zero or more rows with the information of all examinations performed on the specified visit.

10.5 Prescription Service

The Prescription Service provides methods to submit and view prescriptions.

- **Drugs** – Takes no arguments and returns a DataSet from the Drug table that contains information on all drugs currently in the marketplace.
- **OrderPrescription** – Takes 'VisitId', 'DrugId', 'OrderDate', PhysicianId', 'Drug Quantity', 'Dose', 'Refills', 'PrescriptionPurpose', and 'SpecialInstructions'. The method calls a private non-web method called PrescriptionNumberGenerator to create a unique 'PrescriptionNumber'. First, the private method verifies if other prescriptions were ordered during the specified visit. Then, it generates the 'PrescriptionNumber' based on the number of previous orders. A SQL INSERT statement creates a new record with the specified arguments in the PrescriptionOrder table. The method returns a Boolean type that indicates whether the insertion was successful or not.
- **ViewVisitPrescriptionOrders** – Takes a 'VisitId' and returns a DataSet from the PrescriptionOrder table that contains information about all prescriptions ordered during the specified visit.

- **PatientPrescriptions** – Takes a 'PatientId' and returns a DataSet from the PrescriptionOrder table that contains information about the prescription orders for the specified patient.

Again, it is interesting to notice there are two different methods that return prescription information. The first method is targeted specifically for physicians so that they can organize prescription orders by visit. The second method can be utilized by both patients and physicians because it returns a summary of all prescriptions ordered for a particular patient.

10.6 Laboratory Service

The Laboratory service contains methods to submit and view lab test orders.

- **LabTestTypes** – Takes no argument and returns a DataSet that contains a list of all laboratory test types listed in the LabTestType table.
- **OrderLabTest** – Takes 'VisitId', 'LabTestId', 'OrderDate', PhysicianId', and 'Comments'. The method calls a private non-web method called LabTestNumberGenerator to create a unique 'LabTestNumber'. First, the private method verifies if other lab tests were ordered during the specified visit. Then, it generates the 'LabTestNumber' based on the number of previous orders. Finally, a SQL INSERT statement creates a new record with the specified arguments in the LabTestOrder table.
- **ViewVisitLabTestOrders** – Takes a 'VisitId' and returns a DataSet from the LabTestOrder table that contains information about all lab tests that were ordered during the specified visit.

- **PatientLabTestOrders** – Takes a 'PatientId' and returns a DataSet from the LabTestOrder table that contains information about all lab tests that were ordered for the specified patient.

10.7 MedicalTest Service

The MedicalTest service has methods to submit and view medical test orders.

- **MedicalTestTypes** – Takes no argument and returns a DataSet that contains a list of all medical test types listed in the MedicalTestType table.
- **OrderMedicalTest** – Takes 'VisitId', 'MedicalTestId', 'OrderDate', PhysicianId', and 'Comments'. The method calls a private non-web method called MedicalTestNumberGenerator to create a unique 'MedicalTesNumber'. First, the private method verifies if other medical tests were ordered during the specified visit. Then, it generates the 'MedicalTestNumber' based on the number of previous orders. Finally, a SQL INSERT statement creates a new record with the specified arguments in the MedicalTestOrder table.
- **ViewVisitMedicalTestOrders** – Takes a 'VisitId' and returns a DataSet from the MedicalTestOrder table that contains information about all medical tests that were ordered during the specified visit.
- **PatientMedicalTestOrders** – Takes a 'PatientId' and returns a DataSet from the MedicalTestOrder table that contains information about all medical tests ordered for the specified patient.

11 ASP.NET Web Application Design

The final step after designing and coding the Web Services is to create the E-Health Web application. The application ties the user interface with the Web Services.

The E-Health prototype strictly focuses on the patient and physician interfaces. The application is constructed mostly of .aspx pages. It also contains other pages that were developed as standard .html pages with JavaScript. The .html pages are mostly educational and informative, and require no interaction with the user, the Web Services, or the database. The purpose of this section is to present a design overview of the .aspx pages, specifically the pages that interact with Web Services. The discussion will focus on which Web Services methods are invoked and how the data returned by the services is displayed in the application.

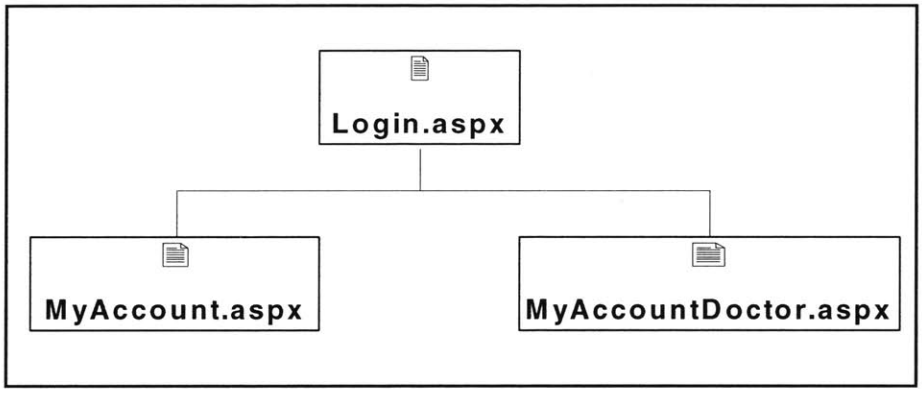

Figure 14. E-Health Login Structure

Figure 14 shows how the prototype application is branched into two sub applications for the patient side and doctor side. The system offers the same login interface for all users, and it is smart to recognize the user's role through the login ID. After successful authorization, the Login.aspx redirects the user to a welcome page according to his role. Once logged in, the application determines who the user is and which pages they can access.

11.1 Patient Application

The architecture of the patient-side application is illustrated in Figure 15. The welcome page for the patient is MyAccount.aspx:

- **MyAccount.aspx** – Invokes the ViewPatientAlerts method, and binds the returned DataSet into a DataGrid for patients to view their personal alerts.

Once in MyAccount.aspx, patients can navigate to five other pages, which are HealthLogs.aspx, MyProfile.aspx, Appointments.aspx, Email.aspx, and MedicalHistory.aspx. HealthLogs.aspx and MedicalHistory.aspx do not use Web Services. The other pages' Web Service usage is explained below.

- **MyProfile.aspx** – Invokes PatientInformation and PatientInsurancePlans, and binds the returned columns to multiple text boxes that display all patient information.
- **Appointments.aspx** – Invokes PatientFutureAppointments to show patients their scheduled visits. The returned DataSet is bounded to a DataGrid control. The page also provides a calendar to select a date for new appointments. Every time a date is selected, the page invokes PhysicianAvailableTimes. The available times are bounded to a drop down list. When a patient selects a date, and clicks on the submit button, the CreateAppointment method is invoked. PatientFutureAppointments is invoked again to display the newly scheduled appointment.
- **Email.aspx** – Includes text boxes that resemble an email format for patients to send email messages to their primary physicians. When the patient clicks on the submit button, the PatientPrimaryPhysician is invoked. The physician's email address is contained in the returned DataSet. The email message is sent to that address.

The HealthLogs.aspx page contains links to six pages that allow patients to enter and track health parameters.

- **WeightLog.aspx** – Invokes ViewWeightLog when the page is loaded. Using the information in the returned DataSet, the page creates a graph that shows the trajectory of all weight measurements. The page provides text boxes for the patient to enter new data. When the submit button is clicked the entered parameters are passed to the UpdateWeightLog method. ViewWeightLog is invoked again to display the newly entered data.
- **ExerciseLog.aspx** – Invokes ViewExerciseLog when the page is loaded. Using the information in the returned DataSet, the page creates a graph that shows the trajectory of all exercise activity. The page provides text boxes for the patient to enter new data. When the user clicks the submit button, the entered parameters are passed to the UpdateExerciseLog method. ViewExerciseLog is invoked again to display the newly entered data.
- **CaloriesLog.aspx** – Invokes ViewCaloriesLog when the page is loaded. Using the information in the returned DataSet, the method creates a graph that shows the trajectory of all calories entries. The page provides text boxes for the patient to enter new data. When the user clicks the submit button, the entered parameters are passed to the UpdateCaloriesLog method. ViewCaloriesLog is invoked again to display the newly entered data.
- **GlucoseLog.aspx** – Invokes ViewGlucoseLog when the page is loaded. Using the information in the returned DataSet, the method creates a graph that shows the trajectory of all glucose measurements. The page provides text

boxes for the patient to enter new data. When the user clicks the submit button, the entered parameters are passed to the UpdateGlucoseLog method. ViewGlucosetLog is invoked again to display the newly entered data.

- **PressureLog.aspx** – Invokes ViewBloodPressureLog when the page is loaded. Using the information in the returned DataSet, the method creates a graph that shows the trajectory of all blood pressure measurements. The page provides text boxes for the patient to enter new data. When the user clicks the submit button, the entered parameters are passed to the UpdateBlood PressureLog method. ViewBloodPressureLog is invoked again to display the newly entered data.
- **PulseLog.aspx** – Invokes ViewPulseLog when the page is loaded. Using the information in the returned DataSet, the method creates a graph that shows the trajectory of all pulse measurements. The page provides text boxes for the patient to enter new data. When the user clicks the submit button, the entered parameters are passed to the UpdatePulseLog method. ViewPulseLog is invoked again to display the newly entered data.

The MedicalHistory.aspx page contains links to four pages that allow patients to view their medical history including conditions, allergies, and ordered tests and prescriptions.

- **HealthConditions.aspx** – Invokes the PatientHealthConditions and PatientFamilyHealthConditions methods and displays the returned DataSets in two separate DataGrids.

- **Allergies.aspx** – Invokes PatientGeneralAllergies and PatientDrugAllergies methods and displays the returned DataSets in two separate DataGrids.
- **Prescriptions.aspx** – Invokes the PatientPrescriptions method and displays the returned DataSet in a DataGrid.
- **Tests.aspx** – Invokes PatientLabTestOrders and PatientMedicalTestOrders methods and displays the returned DataSets in two separate DataGrids.

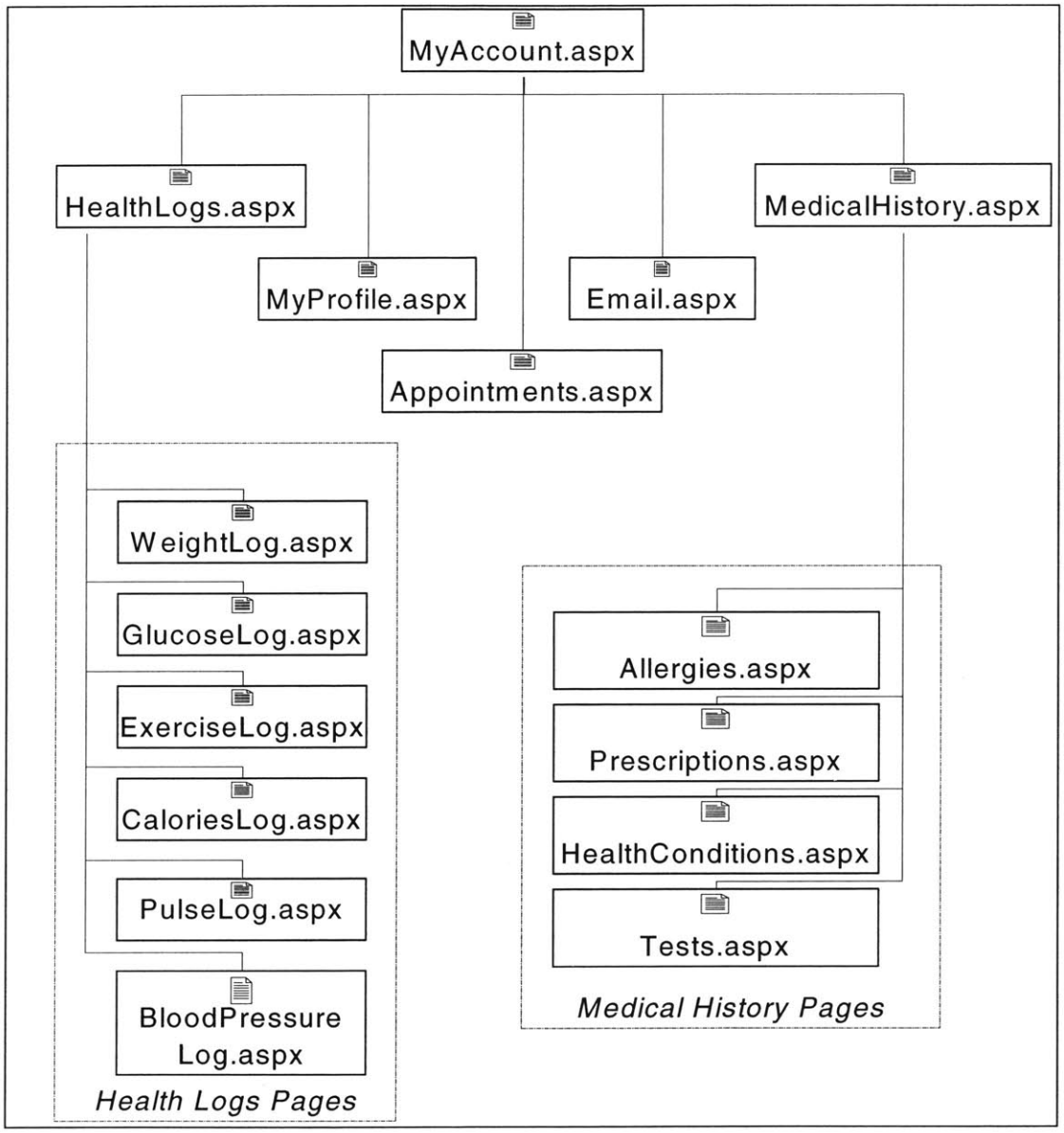

Figure 15. Patient Application Architecture

11.2 Physician Application

The architecture of the physician-side application is illustrated in Figure 16. The welcome page for the patient is MyAccountDoctor.aspx:

- **MyAccountDoctor.aspx** – Invokes the ViewPhysicianPatientAlerts method, and binds the returned DataSet into a DataGrid. The PhysicianPatients method is also invoked and is bounded to a drop down menu that displays a list of the physician's patients. The page also provides a calendar to select a date for viewing appointments. Every time a date is selected, the page invokes PhysicianAppointments. The returned DataSet with the appointments for the selected date is bounded to DataGrid.

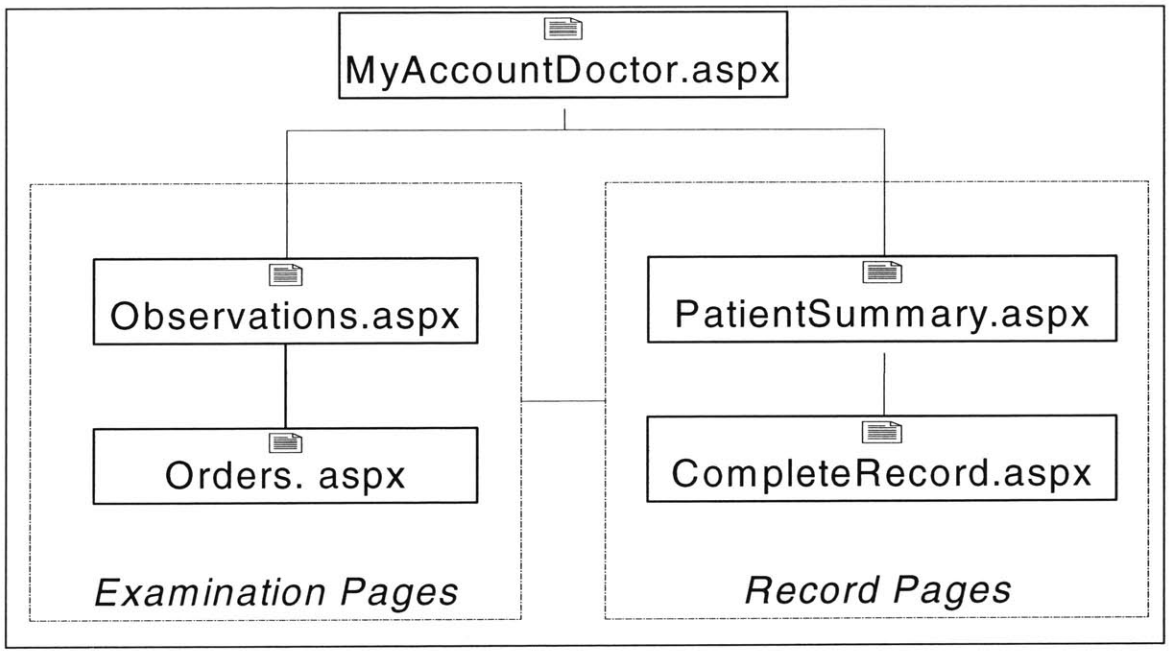

Figure 16. Physician Application Architecture

Once in MyAccountDoctor.aspx, physicians have the option to examine a patient that has a scheduled appointment on the current date. This is done by selecting one of the

patient's in the appointment's DataGrid control. Two pages allow physicians to record visit-related information.

- **Examination.aspx** – The page invokes the ExaminationTypes methods, and binds the DataSet to a drop down list that will show physicians all available types of examination. The physician can make a selection, enter data in various text boxes, and when the submit button is clicked the page invokes CreateExamination. The page shows a confirmation message according to the returned Boolean type.
- **Orders.aspx** – The page invokes Drugs, LabTestTypes, MedicalTestTypes, and binds the three returned DataSets to three drop down lists. Physicians can select a drug from the list and submit a prescription order. The page invokes the OrderPrescription when the physician makes the selection. The same applies for lab tests and medical test orders. In those cases, OrderLabTest and OrderMedicalTest are invoked, respectively. The page shows a confirmation message in all three cases according to the returned Boolean type.

On MyAccountDoctor.aspx, physicians also have the option of viewing a patient's record without having to examine that patient. This is done by selecting one of the patients in the drop down patient list. Two pages contain record-related information.

- **Record.aspx** – When the page loads, the PatientPastAppointments method is invoked. The DataSet is bounded to a drop down list that shows the date of all past appointments. Physicians can select any of the past visits from the list to view detailed visit information. When a particular visit is selected, the page invokes ViewVisitExamination, ViewVisitLabTestOrders,

ViewVisitMedicalTestOrders, and ViewVisitPrescriptionOrders. The DataSets returned by the four methods are bounded to four different DataGrids.

- **PatientSummary.aspx** – The page invokes seven methods, all related to the health logs tables. These are ViewBloodPressureLog, ViewGlucoseLog, ViewTemperatureLog, ViewCaloriesLog, ViewPulseLog, ViewExerciseLog, and ViewWeightLog. The page uses the returned DataSets to display graphs with the trajectory of each health measurement.

12 E-Health Future Development Cycles

The E-Health prototype that this document presents implements a portion of the envisioned business model, and, thus, needs to undergo further development cycles to provide full value to all stakeholders. Before continuing with the next iteration, the E-Health prototype needs to be market tested with representative users. Testing allows (1) gathering user performance data, including types of errors and satisfaction levels; (2) identifying particular areas of the interface that presents problems to the user; and (3) exploring new product functionalities. At this stage of development, changes are still easy and cheap. The team must assure performance optimization, and most importantly, make sure users feel comfortable with the system. Testing results will provide the basis for the requirements of the next iteration of the spiral development model. The rest of this section will cover potential expansion areas that should be targeted in future cycles.

12.1 Data Mining Web Services

Data mining uses large data sets to construct statistical models that aid in classifying and predicting new observations. The E-Health Network can easily gather

anonymous data from all health care institutions that are affiliated to the network. This provides richer data to construct models that are more robust.

Web Services play an important role in collecting the distributed data, constructing the models, and propagating the model results to affiliated institutions. Gathered data can be arranged in vectors of information. These vectors can include patients' demographics, health measurements, blood test results, prescriptions, and past treatments. Data mining techniques can construct models with the existing vectors to classify other patients' risk and to suggest the appropriate medical action. In addition, it is possible to cluster patients into groups to make more accurate predictions of their prognosis. The database structure must be modified and new Web Services must be constructed to provide data mining functionalities.

12.2 Increased Portal Personalization

E-Health offers personalized views depending on a user's role, but offers the same interface for all users of a certain role. This means that all physicians accessing the system will see the same format of information. Flexibility is a key issue in the health care industry, and the system should provide greater flexibility in its application. For example, each physician should be able to choose what he or she wants to see in their portal by selecting and arranging the displayed objects. This is similar to existing Internet portals that allow users to select components of interest. Web-based applications that provide a true separation of data and presentation, like E-Health, offer a number of different views of the same data. Thus, the technologies E-Health uses will facilitate the personalization functionality.

12.3 Data Sharing Permissions

A benefit from the E-Health Network is the ability to share patient data with physicians in the same hospital and at other distant health institutions. This facilitates the physician referral process, and allows physicians to have a comprehensive view of a patient's history. Physicians are not limited to data from their particular relationship with a patient, but can extend their analysis to include the patient's other medical interactions and relationships. Due to the sensitive nature of health care data, this area needs a lot of legal work, and all involved parties must agree to release information. On the technical side, implementing security protocols is vital to restrict information access only to parties with permission. In addition, this would include adding new database tables and new fields to existing tables. New web methods must be developed to create and verify data permissions, and to share data between the different parties.

12.4 Educational Communities Development

Communities have become an important concept in the Internet because they create value by bringing together people with similar interests and needs in a virtual space. The creation of an E-Health online community will provide an easily accessible environment for learning and sharing medical knowledge with a broad community of experienced caregivers and patients. Physicians belonging to the E-Health Network can be grouped by their specialization or research area. Through their community space, they can share the latest discoveries in their areas, discuss data mining results, and share opinions about patient treatments. Patients can be grouped by their health conditions to share their views on a particular subject, discuss the latest health news, and discuss their symptoms and fears with others in a similar situation. The community space should also

allow interaction and discussion between the patients and physicians. These features could be implemented through message boards, chat rooms, forums, and seminars.

12.5 Expand and Enhance Interfaces and Services

The vision of the E-Health system is that any activity that takes place in a health care institution can be completed through the system. For this to become a reality, E-Health needs to enhance its current offering and expand into new areas. It will be necessary to create personalized portals for other system actors, in addition to improving the existing patient and physician portals. Other system actors include hospital administrators, nurses, paramedics, lab technicians, and pharmacists. Market testing will help understand what are the requirements for each group. Example of new potential features include:

- Lab tests results insertion by lab technicians
- Lab test results viewing by physicians
- Prescription management tools for pharmacists
- Hospital administration tools to manage personnel and patient relationship
- Management of hospitalization and emergency visits
- Recording surgery and medical test information including graphs and video
- Adaptive alerts that allow quick physician response
- Alerts for lab and medical test results
- Personalized education tools

New roles and functionalities imply addition of Web Services methods, database changes, and creation of new interfaces. Certainly, some of the existing methods can be used with the new interfaces. In terms of the database, it is important to point out that the

E-Health model was constructed specifically for the E-Health application prototype, and does not take into account existing legacy databases at any health care institution. The team did research this area, but for time-constraint reasons decided to create an independent model. A real implementation must assure compatibility with existing systems. In addition, it is necessary to make information more even across the different database areas. Currently, some groups of data have richer and more flexible data structures than others. Migration to XML and object-oriented databases should be carefully examined.

13 E-Health Development Lessons

This project certainly proved to be a learning experience. The opportunity of working with new technologies was constructive, but the opportunity to experience a software development process was most valuable. It is vital to implement a rapid development model that provides flexibility to quickly adapt to changes in requirements. The model must also manage the complexity of a project by reducing development risks. Team organization and motivation are also key factors to a successful implementation.

It is necessary to have a clear vision and understanding of the market and its stakeholders, including important relationships between them. Any new product development must be structured around its value proposition to all stakeholder segments. The product core benefits must be aligned with their needs. Technology for technology's sake may attract early adopters and enthusiasts, but products with a misaligned value strategy will rarely reach mainstream customers.

In the past years, technologists with high hopes believed that a business could be based solely on a cool technology. They thought technology was the difficult part of

creating a new business, and that it was the exclusive source of competitive advantage. When the Dot Com bubble burst, it became evident how far they were from reality. New technologies can unquestionably help build innovative business models, but technology alone cannot create a sustainable model. A technology platform can certainly contribute to competitive differentiation, but with a high availability of technologies and skilled people, it should not be a business' main source of competitive advantage. Only technologies that are strategically interlinked with the business components are those that can form the basis of competition advantage.

The E-Health team understood this concept, and strived to grasp a comprehensive view of the health care industry in spite of development time-constraints. In a real business environment, more time should be spent learning about the world being modeled because it will prove to be advantageous in the system design phases. An intelligently organized team, with a strong understanding of the business environment and access to innovative technologies, is the foundation of any successful new product development.

14 References

Matheson, David, and Tom Robinson. "Disease Management Takes Flight." 2000. The Boston Consulting Group Health Care Practice. 22 Oct. 2001. <http://www.bcg.com/publications/search_view_ofas.asp?pubID=570>.

McConnell, Steve. *Rapid Development.* Microsoft Press, 1996.

Newman, Heather. "Patients probe Internet for Information" Detroit Free Press 3 Aug. 1999. Technology. 1 Nov. 2001 < http://www.freep.com/tech/qjoy3nf.htm>.

Ramirez, Lisa. "The Internet is flooded with medical information, but what should you believe?" Detroit Free Press 3 Aug. 1999. Technology. 1 Nov. 2001 <http://www.freep.com/tech/qmedweb3nf.htm>.

Robinson, Simon. *Professional C#.* Wrox Press, 2001.

Saunders, Norman C. and Betty W. Su. "The U.S. economy to 2008: a decade of continued growth ." Nov. 1999. Bureau of Labor Statistics. 15 Oct. 2001. <http://www.bls.gov/opub/mlr/1999/11/art2full.pdf>.

Walker, David M. "Medicare: New Spending Estimates Underscore Need for Reform." 25 July 2001. United States General Accounting Office. 15 Oct 2001 <http://www.gao.gov/new.items/d011010t.pdf>.

Walther, Stephen. *ASP.NET Unleashed.* Sams Publishing, 2002.