A WEB-ACCESSIBLE RELATIONAL DATABASE FOR INTACT ROCK PROPERTIES AND AN XML DATA FORMAT FOR INTACT ROCK PROPERTIES WITH SCHEMA

by

Alexandr Turichshev

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science
Graduate Department of Civil Engineering
University of Toronto

© Copyright by Alexandr Turichshev (2002)
A Web-Accessible Relational Database for Intact Rock Properties and an XML Data Format for Intact Rock Properties with Schema

Master of Applied Science, 2002

Alexandr Turichshev

Department of Civil Engineering, University of Toronto

Rock engineering is well-known for dealing with limited information. Geotechnical investigations are expensive. During preliminary design stages, it is common to estimate design parameters based on data from other investigations around the same area or on data from testing similar rock types. Finding such data can be challenging.

This thesis attempts to provide a solution to the existing situation. The author has developed an extensible relational database for intact rock properties. The database can be accessed through the Internet via a web-based interface. The database and the interface together form “RockProp”, a web-based application. The application provides a widely-accessible point of reference for intact rock engineering data.

The thesis also presents the development of a flexible XML format for intact rock properties together with XML schema. XML is a promising and rapidly growing electronic data exchange format. The schema was developed to control its use: its description and enforcement.
ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr. John H. Curran, for his help throughout the course of this research work. Successful completion of this project would not be possible without his guidance and expertise. His organization of exceptional financial support was a key component for beginning and finishing this work.

I wish to thank my two officemates, Reginald Hammah and Vijayakumar Sinnathurai, for long intellectual conversations. I want to acknowledge people at Rocscience Inc. for their criticism of my work. Nobody is perfect.

I would like to thank my parents and my brother for giving me energy during this long period. Their continuous support was essential. I thank them for making me feel special, for believing in me.

I am very grateful to my girlfriend, Nadine. Her continuous support, encouragement, patience, understanding, and love during the last 2 years were invaluable. She made it possible to complete this thesis within the timeframe required.

I would like to thank Nadine’s parents, Ivor and Nchamah Miller, for being in my life, and for their encouragement and support.
Finally, I would like to thank people who helped me with editing of my thesis: Nadine Miller, Ivor Miller, Dr. John Curran, and Dr. Reginald Hammah. Their work is very much appreciated.
TABLE OF CONTENTS

A Web-Accessible Relational Database for Intact Rock Properties And an XML Data Format for Intact Rock Properties with Schema ........................................................................................................... i

Abstract .......................................................................................................................................... ii

Acknowledgements....................................................................................................................... iii

Table of Contents ........................................................................................................................... v

List of Tables ................................................................................................................................ xi

List of Figures .............................................................................................................................. xii

List of Appendices ....................................................................................................................... xv

CHAPTER ONE – INTRODUCTION .......................................................................................... 1

1.1 Project Objectives ............................................................................................................... 2

1.1.1 Immediate Objectives ................................................................................................. 2

1.1.2 Subsequent Objectives ............................................................................................... 3

1.2 Addressing Specified Objectives .................................................................................... 3

1.3 Conclusion........................................................................................................................... 5
CHAPTER TWO – ROCKPROP DATA ...................................................................................... 6

2.1 Selection of Data Parameters .......................................................................................... 6

2.2 Data Sources.................................................................................................................... 7

2.3 Parameter Organization and Data Structure .................................................................... 9

CHAPTER THREE – ROCKPROP’S STRUCTURE ................................................................ 12

3.1 3-Tier Architecture ........................................................................................................ 12

3.1.1 3-Tier vs. 2-Tier Architecture ................................................................................ 13

3.1.2 Server-Centric and Client-Centric Approaches ..................................................... 14

3.1.3 RockProp................................................................................................................ 15

3.2 Server Tier ..................................................................................................................... 16

3.2.1 RDBMS.................................................................................................................. 16

3.2.2 Database Objects: General to All Packages and Specific to SQL Server ............. 18

3.2.3 Main Database: RockProp ..................................................................................... 20

3.2.3.1 Structure.......................................................................................................... 21

3.2.3.2 Table Layout, Primary and Foreign Keys, Relationships...................................... 21

3.2.3.3 Other Database Objects: Stored Procedures, Views, and Roles......................... 26
3.2.4 Satellite Database One: GeoRef

3.2.4.1 Structure: Table Layout, Primary and Foreign Keys, Relationships

3.2.4.2 Other Database Objects: Stored Procedures, Views, and Roles

3.2.5 Satellite Database Two: RPUUsers

3.2.5.1 Structure: Table Layout, Primary and Foreign Keys, Relationships

3.2.5.2 Other Database Objects: Stored Procedures, Views, and Roles

3.3 Client and Middle Tiers

3.3.1 Client Tier

3.3.1.1 ASP Technology

3.3.1.2 Web Browser

3.3.2 Middle Tier

3.3.3 Interaction Mechanisms Between Web Application and Middle Tier and Between Middle Tier and Database Server

3.4 Physical Location of Components

CHAPTER FOUR – ROCKPROP’S FUNCTIONALITY AND ITS IMPLEMENTATION

4.1 The RockProp Web Interface Functionality
4.2.3.3 Geo-Region Search ................................................................. 52
4.2.3.4 Country Search ................................................................. 53
4.2.3.5 Feature Search ................................................................. 54
4.2.4 RockProp Database Data Searches ........................................ 58
4.2.5 Displaying Record Information .............................................. 59

4.3 Requirements, Installation, and Configuration of RockProp .......... 62

CHAPTER FIVE – XML DATA FORMAT AND XML SCHEMA FOR THE INTACT ROCK MECHANICS DATA ......................................................... 64
5.1 XML and Its Features ............................................................... 65
5.2 XML Schema and Its Role ....................................................... 68
5.3 Benefits to Geotechnical Industry ............................................. 70
5.4 Relevance to RockProp ............................................................ 72
5.5 Design Requirements .............................................................. 72
5.6 Design of the XML Data Format and the Schema ....................... 72
5.6.1 Geotechnical Data Structure ............................................... 73
5.6.2 XML Document Format Design ......................................... 74
LIST OF TABLES

Table A.1: List of database rock strength parameters………………………………………A-2

Table A.2: List of other database parameters………………………………………………A-3

Table C.1: List of geo-features and their codes……………………………………………C-3

Table C.2: List of geo-regions, their codes, and database table names……………………C-3

Table C.3: List of countries, their codes, and geo-region codes they are listed under……C-4
LIST OF FIGURES

Figure 2.1: Data hierarchy ................................................................. 11

Figure 3.1: General structure of the RockProp project ................................. 16

Figure 4.1: Web application’s folder structure ............................................. 44

Figure 4.2: Frame structure of the data output window ............................... 60

Figure 5.1: Example of XML data ......................................................... 65

Figure 5.2: Example of XML data structure ............................................. 66

Figure 5.3: Example of XML extensibility ............................................... 67

Figure 5.4: Structure of geotechnical data used for XML data format design .... 73

Figure 5.5: Skeleton of the XML document ............................................. 74

Figure B.1: RockProp database table layout ......................................... B-2

Figure B.2: Layout of tables and table columns of the RockProp database (main section) .... B-3

Figure B.3: Layout of tables and table columns of the RockProp database (sub-record section) ................................................. B-4

Figure B.4: Layout of tables and table columns of the RockProp database (reference section) ................................................. B-5

Figure C.1: Layout of tables and table columns of the GeoRef database ........... C-2

Figure D.1: Layout of tables and table columns of the RPUsers database .......... D-2
Figure E.1: *RockProp’s* startup screen

Figure E.2: User registration form

Figure E.3: Login form

Figure E.4: *RockProp’s* welcome screen

Figure E.5: Searching *RockProp* by rock type: rock type input (a), search results (b)

Figure E.6: Searching *RockProp* by geographical coordinates: coordinate and search radius input (a), search results (b)

Figure E.7: Searching *RockProp* by geographical region: region selection (a), search results (b)

Figure E.8: Searching *RockProp* by country: region selection (a), country selection (b), search results (c)

Figure E.9: Searching *RockProp* by feature: selecting geographical region (a), selecting country (b), selecting first order administrative division (c), selecting feature type, feature name and search radius input (d), selecting a particular feature from all matching the search criteria (e), search results (f)

Figure E.10: Viewing record information: general information (a), intact index properties (b), reference information (e)

Figure H.1: Root element (*<geotechnicalData*>) of the document with its immediate children: *<location>*, *<rockMass>* and *<references>*

Figure H.2: *<rockMass>* element has one child element *<dataCategory>*. *<dataCategory>* contains four children: *<location>*, *<depth>*, *<materialDescription>* and *<properties>*
Figure H.3: Material is described by `<rockType>`, `<fabric>`, and `<unitWt>` that contain additional tags

Figure H.4: `<properties>` element contains several `<property>` elements each corresponding to a particular test or procedure. Properties include compressive strength, tensile strength, acoustic velocity, Hoek-Brown constants, Schmidt hammer, point load, fracture toughness, slake durability

Figure H.5: Elements related to compressive strength

Figure H.6: Elements related to tensile strength

Figure H.7: Elements related to acoustic velocity

Figure H.8: Elements related to Hoek-Brown constants

Figure H.9: Elements related to Schmidt hammer data

Figure H.10: Elements related to point load data

Figure H.11: Elements related to fracture toughness data

Figure H.12: Elements related to slake durability data

Figure H.13: Elements related to reference data
LIST OF APPENDICES

APPENDIX A: Project Parameters.................................A-1 – A-3
APPENDIX B: RockProp Database Structure and Objects.............B-1 – B-98
APPENDIX C: GeoRef Database Structure and Objects...............C-1 – C-67
APPENDIX D: RPUsers Database Structure and Objects...............D-1 – D-17
APPENDIX E: RockProp Web Application Interface.....................E-1 – E-13
APPENDIX F: RockProp Web Interface Code..........................F-1 – F-63
APPENDIX G: Middle Tier Classes........................................G-1 – G-33
APPENDIX H: XML Data Format Structure and Examples...............H-1 – H-10
APPENDIX I: XML Schema..................................................I-1 – I-14
CHAPTER ONE – INTRODUCTION

Data… Every engineering decision in every country is based on some kind of information: technical, financial, social, etc. Information is based on data. The amount of data often affects the quality and reliability of information, and thus influences the judgement of an engineer. Data has little value if it is not readily found in a useable or accessible form. The value of data significantly increases if it can be accessed easily and rapidly. The availability of numerous search methods, capability to sort data, recognition of its basic structure and relationships among components, simplifies access to the data and, therefore, increases its value. Today, with low computer costs and development of powerful inexpensive database packages, electronic databases provide the only realistic approach in achieving those goals.

Rock engineering is well-known for its need to deal with data-limited problems. Rock strength testing is expensive, time-consuming and labour-intensive. Samples have to be obtained from a project site, which is often located in another country, on another continent. Eventually, a company responsible for a project’s design will conduct necessary geotechnical investigations. During a preliminary design stage, however, clients are not usually prepared to spend significant amounts of money on such investigations. It is common in the geotechnical engineering practice to estimate initial design requirements on the basis of already available information: results of other geotechnical investigations from the same area, strength parameters of similar
rock types. The ability to easily access such information is a key to success during the initial stages of any geotechnical project.

This M.A.Sc. thesis attempts to provide a solution to the data accessibility problem. The subject for this research was proposed by Dr. J. H. Curran who saw a necessity for, and the potential convenience of, a single easily-accessible point of reference that would provide required data for users around the world. *RockProp* (Rock Properties) is an Internet-accessible electronic database application that was created with the intention of becoming such a reference and was developed by the author under the supervision of Dr. Curran.

1.1 **PROJECT OBJECTIVES**

The main objectives established for this thesis can be divided into two categories: immediate objectives (established before project work had commenced) and subsequent objectives (established during the project’s development).

1.1.1 **Immediate Objectives**

Four main objectives were established before project work had started. First, an initial set of data parameters to be implemented in the database had to be selected from parameters related to rock engineering. This was a necessary step; it would be impossible to achieve meaningful results within the timeframe allocated for completion of the M.A.Sc. degree if every aspect of rock mechanics was considered.

Second, a hierarchical structure needed to be developed for selected parameters. This was crucial for further work. The development of a hierarchy within data would provide enormous
help in the database design stage, simplify and accelerate data searches in the future; it would also allow future incorporation of additional parameters into the database structure. The development of a proper data organizational structure was essential for ensuring extensibility of the project.

The design and implementation of a relational database for selected parameters was the third objective. The database would become the core of the project, storing, managing, and controlling data and its access.

Finally, in order to provide database access to the general public, an application would have to be developed. The fourth immediate objective was development of an application capable of data searches, extraction and new data input. Its main requirement would be to provide database access from any part of the world.

1.1.2 Subsequent Objectives

Two main objectives were added as work on the project progressed. Development of an XML (Extensible Markup Language) schema based on the existing (developed at the time) database was proposed by Dr. Curran. The schema would have to be based on an XML document. Therefore, the XML format for the data had to be developed prior to the development of the schema.

1.2 Addressing Specified Objectives

At the completion of this thesis all specified objectives were achieved. A number of intact rock mechanics parameters were selected for initial implementation in the database. The selected
parameters are those regularly used in analyses of rock engineering problems. They were grouped according to standard tests used to determine them. Since the intact testing is performed on samples, the concept of sample testing was used to organize the parameters. An in-depth description of parameter selection criteria and data organization is the subject of Chapter 2.

The database for the data storage and management was designed on the basis of the developed data structure. Several new parameters were introduced during the database design stage. In order to relate selected parameters to the physical origin of data, geographical coordinates were included. A description of rock type was also added. Chapter 3 contains the detailed description of the database design and implementation along with an overview of the entire project structure.

Today, the Internet is the most economic and widely used means of communication worldwide. Naturally, the ability to access the database via the Internet using a web browser would significantly simplify and accelerate access to data and thus vastly increase the number of potential users. The Internet-access rationale determined the application development path. The application was designed and developed so that a web browser is used for database access, information searches and retrieval, rudimentary user statistics and data input. The application’s structure and functionality are described in detail in Chapters 3 and 4.

Finally, an XML data format and XML schema were developed. The existing database structure was used as a guide for the XML document design. The schema to which the document conforms was then implemented. Chapter 5 of the thesis describes the approach and presents how the subsequent objectives were achieved.
1.3 Conclusion

The result of this work is *RockProp*. The author would like to call it a work in progress believing that his work is only the beginning. *RockProp* is a combination of the database and a front-end application for accessing rock engineering data through the Internet.

The final portion of this document presents recommendations for the future development of this project and proposes some possible changes that might improve the project overall.
CHAPTER TWO – ROCKPROP DATA

The centre of every database application is the data itself. Data is the only reason why databases exist and, therefore, is highly important. In order for RockProp to initially attract people initially and make sure that they use it in the future, the database must contain a useful set of parameters. The data has to be reliable since there is very little use for data that is not trustworthy. Selection of parameters and determination of appropriate data sources was the first important stage of the project.

2.1 SELECTION OF DATA PARAMETERS

The selection of data parameters was performed in three stages. First, a number of essential parameters were obtained from a discussion with several rockmechanics specialists. Three Lassonde Mining Institute researchers (Dr. W. Bawden, Dr. J. Curran, and Dr. B. Mohanty) and three outside specialists (Dr. J. Carvalho of Golder Associates, Dr. M. Diederichs of Queens University, and Dr. B. Corkum of Rocscience Inc.), as well as the author, met to discuss which parameters should be initially included in the database initially. In addition, the participants discussed reliable sources of data. One source (Lama et al., 1978) was mentioned by several people as an appropriate starting point. This book became the first reference point.

The second stage of parameter selection was directly related to the book suggested in the first stage (Lama et al., 1978). Along with an extensive amount of data, this book contained
references to published papers from which the data was obtained. After becoming familiar with a number of references, the author added several parameters to the ones obtained during stage 1. These parameters are often acquired during the intact strength testing of rock. Furthermore, several parameters were adopted from a second reference (Jackson et al., 1995) suggested by Dr. W. Bawden.

Besides the parameters directly related to rock strength from stages 1 and 2, a number of additional parameters were incorporated by the author. This comprised stage 3 of the selection process. The additions were mostly related to material description, physical source of samples (geographical location), and data referencing (sources of data). Stage 3 was the final stage of the selection process.

A complete list of parameters used in the database is provided in Tables A.1 and A.2 (Appendix A).

### 2.2 Data Sources

At present, there are three data sources for the RockProp database. Two of them were mentioned in Section 2.1: books by Lama et al. (1978) and Jackson et al. (1995). The third source consists of published papers that were referenced in the book by Lama et al. (1978).

The first book (Lama et al., 1978) provides over 2000 records (in total) from many countries on various rock types. Documented parameters include: rock type, density, modulus of elasticity, modulus of rigidity, Poisson’s ratio, compressive and tensile strengths, and various remarks related to rock description.
Although a useful source, this reference has two disadvantages. First, even though it lists the geographical source of material for every record, it does not provide geographical coordinates. Before these records can be entered into the database, appropriate geographical coordinates have to be determined. This has to be done manually, searching geographical atlases or the Internet. Even then, obtained coordinates do not designate a sample location but rather the location of a project or an area (dam, mine or basin) the data originated from. Such records, therefore, must be used with caution. Database users must pay attention to the geology of the area; they can rely primarily on the description of rock type and use coordinates for global reference only. The second disadvantage of this reference is related to data records themselves. Many entries do not contain data relevant to a number of parameters that the book references. This is not a problem when some non-essential parameters are missing. Some records, however, lack entire sets of important parameters like compressive and tensile strengths, the modulus of elasticity, and Poisson’s ratio. These parameters are essential in rock engineering, especially during the preliminary design stage. Such records could not be used in the database at all. Other than these shortcomings, the book by Lama et al. (1978) is a valuable reference.

The second reference (Jackson et al., 1995) is outstanding. The book contains data for a number of parameters: sample origin (including geographical coordinates), rock type, important mechanical parameters (Young’s modulus, compressive and tensile strengths, sample dimensions, etc.), and Hoek-Brown material constants. Unlike the previous one though, this reference only contains 77 records, all from Canadian mines. Having a smaller number of records and lacking the extensive spatial coverage of the first reference, this book, however, contains a broader spectrum of data for almost every entry. It is an invaluable resource.
The third source is related to the first reference (Lama et al., 1978) and should be regarded as being complimentary to it. In order to obtain detailed information (sample origin and description of testing methods) for as many records as possible, a great deal of effort was invested in finding every available reference for the records listed by the book’s authors. All credit for this work is given to Kien Vinh Duong, a summer student who was employed by Dr. Curran in 2000. Kien assisted in finding references. Approximately 100 journal articles (out of 220 listed) will become a complimentary contribution to the data in the book by Lama et al. (1978).

So far the author has talked about references that can be readily used in the database today. The extensible design of the database has the ability to incorporate other reference sources, e.g. data supplied by geotechnical consulting and mining companies. Without doubt, these companies have enormous data resources accumulated over many years. Such data would greatly add to the usefulness of RockProp. It is expected that the data from the industry will become an important part of the project.

2.3 **Parameter Organization and Data Structure**

The issue of data hierarchy may not be important when working with a small number of parameters. When the number of parameters is large and one of the objectives is to allow easy addition of new ones, the issues of hierarchical data structure and parameter organization become important. Development of hierarchy aids with the database design process, allows one to focus on specific properties during data searches, simplifies and accelerates these searches, and therefore, is invaluable for the development of the entire project.
The subject of parameter organization was briefly touched upon before. The data parameters related to geotechnical laboratory testing were separated into corresponding groups, established according to the test type or procedure. Every parameter in each group is either obtained by a particular testing method (for example, compressive strength) or related to the testing procedure itself (for example, core dimensions or confining stress range). Besides being simple, this organization approach allows for the easy additions of new parameters. For example, if it is decided to incorporate data for some additional type of testing in the future, a new group can be easily created without affecting ones that already exist. Organization of intact parameters into groups can be seen in Table A.1 (Appendix A).

The reference information for data is organized in a similar manner; parameters are grouped according to reference type: book, conference proceeding, or journal article. This organization allows for new reference types to be easily incorporated; the addition of a new reference source will not affect ones which are already implemented. Existing reference parameters and their organization can be found in Table A.2 (Appendix A).

The entire data structure and hierarchy is shown on Figure 2.1. This figure shows how data is structured, and where the aforementioned groups fit in. The record represents the root of the hierarchy. The record is a tested sample or a set of samples that have the same rock type and spatial origin. Each record is described by location and material description. A record also includes reference information and the source of the data.

Every record contains one or several sub-records. Each sub-record corresponds to one of the intact test groups. It represents information related to a single test type. For classification purposes, each sub-record is described to be of a particular data type and data source. The data
type classifies the sub-record on a global scale of data origin. For example, only intact parameters are currently considered; therefore, every sub-record has intact data type classification. Other future data types may include in-situ data, structural data, etc. The data source represents the next level of classification refinement; it determines to which group (test method) parameters belong. There are eight data sources at present; they are shown in Figure 2.1.

The author would like to point out that the presented data hierarchy is closely related to the database design which is presented in the next chapter.
CHAPTER THREE – ROCKPROP’S STRUCTURE

This chapter presents the architecture of the entire RockProp project. It provides some background information on client/server computing and database design. It gives an overview of databases built for the project, web application, and interaction mechanisms among the project’s various components.

3.1 3-TIER ARCHITECTURE

There are two types of client/server architectures in use today: 2-tier and 3-tier. The majority of business database applications in use today were built using a 2-tier model. Applications currently being developed mostly have 3-tier architecture. The RockProp project uses a 3-tier client-server architecture model.

A 2-tier client/server application consists of a database server and one or several types of client applications. The database server hosts data. A client application is used to access the database server; it sends requests to the database. The database server responds, extracts data, and returns results to the client application. This is the simplest model of interaction between a database and a client. The reason why this model has been popular in the past is its ease of design and implementation (Williams, 1999, p.26).
The 3-tier client-server application consists of the database server, the middle tier, and the client application(s). Data is still hosted by the database server; the client, however, never accesses it directly. Every client’s request is addressed to the middle tier instead. The middle tier passes requests to the database server. The database server passes results to the middle tier which then passes them to the client. Design and implementation of this architecture is more complex than that of the 2-tier model but has a number of advantages over it (Williams, 1999, p.27-29).

3.1.1 3-Tier vs. 2-Tier Architecture

The 3-tier architecture model allows easy component updates (Vieira, 1999, p.18). For example, suppose that a database access procedure is located in the middle tier instead of the client. When this procedure is modified, only its one instance is updated in the middle tier instead of multiple procedures being updated on each client. This eliminates need for multiple updates of components.

The 3-tier architecture also facilitates component reuse. For example, assume there is more than one client type (Internet and local based) accessing the data on the server. The 3-tier architecture allows use of the same data access component for both client types; there is no need to develop a specific component for each type of the client.

As the number of users accessing the data server increases, the number of database connections in the 2-tier application increases as well, and thus slowing the database server. In the 2-tier architecture, the database connection remains open until the client receives a response from the database server. The 3-tier architecture can reduce the number of connections by quick connection closing or connection sharing. Some database connections can be closed quickly in
the middle tier before passing the request to the client. Open connections can also be shared with the use of resource pooling (Williams, 1999, p.29).

The 3-tier model provides greater data security than the 2-tier one (Williams, 1999, p.28). The middle tier makes it hard to trace how the database data is being accessed. This is especially crucial when Internet-based clients are used.

Finally, if the number of the data servers increases, the middle tier can provide seamless connections to multiple servers; the use of the 2-tier model in this case would require re-design of client applications (Williams, 1999, p.27).

3.1.2 Server-Centric and Client-Centric Approaches

Two architecture subtypes are recognized in client/server computing (in both 2- and 3-tier architectures): client-centric, also called fat client, and server-centric, also known as fat server (Vieira, 1999, p.15, 16).

The client/server model is client-centric when the majority of logic (data manipulation, decision making) is handled by the client. The data server only responds to incoming requests and returns results to the client. Such an approach is often characterized by higher network traffic and difficult updates of multiple client components. At the same time, this approach places fewer demands on the data server (Vieira, 1999, p.16; Williams, 1999, p.22).

The server-centric approach is characterized by a client application being responsible mostly for presentation-related functions. The application logic is performed entirely on the server. Advantages of this model include less network traffic, little demand on client computers
allowing presentation to be improved, and uniform updates of some modules for all clients (server-side updates) to be made. In order to be effective, the fat server model may require a more powerful server (Vieira, 1999, p.17; Williams, 1999, p.22).

Another possibility in the 3-tier architecture arises when application logic is located on the middle tier. This is the combination of the two approaches; interaction between the client tier and the middle tier is the thin client approach, and between the middle and the server tiers – fat client approach. Depending on the location of the middle tier and its built-in functionality, it is possible to obtain the best characteristics of the two approaches.

### 3.1.3 RockProp

RockProp’s architecture employs a 3-tier model with application logic spread between the server and the middle tiers. In the Internet accessible scenario, with the web browser being the client and network traffic being an issue, it is almost impossible to implement the client-centric approach effectively. Even though today’s browsers can perform some computing (using *JScript*, *VBScript*, or *JavaScript* languages), their capabilities are still very limited.

Additionally, inherent implementation differences between different browsers would prevent results from being uniform across browser brands and different computer platforms. The best solution is to let browsers do what they do best: render HTML, display information on the screen, and thus primarily perform only presentation functions.

The issue of information presentation in RockProp is a little more complex; the presentation functionality is split between the user’s browser and the ASP (*Active Server Pages*, Microsoft’s technology for dynamic web pages) engine on the web server.
RockProp’s logic is spread between the middle and the server tiers. Such an approach enables flexibility in the implementation and future maintenance of the entire application. Figure 3.1 shows RockProp’s general structure.

The remainder of this chapter provides a detailed description of the application’s implementation.

![Figure 3.1 General structure of the RockProp project.](image)

### 3.2 Server Tier

The main element of RockProp’s server tier is the Relational Database Management System (RDBMS) hosting three databases: RockProp, GeoRef, and RPUsers.

#### 3.2.1 RDBMS

A database is simply a collection of related data. A filing cabinet is an example of a simple database. A database management system (DBMS) is a computer application for managing
access to a database. It provides an environment for convenient and efficient data storage and retrieval (Bonner, 2001, p.1-2). When the amount of data in a database is small, it can often be sufficiently managed in one table (flat database). A simple word processor or a spread sheet application are sufficient to fulfill the role of the DBMS in this case. When the amount of data is large, it has to be divided among multiple tables in order to eliminate data redundancy, update, insertion and deletion anomalies (more on these in the next section) (Roman, 1997, p.6, 7). Such databases are called relational because information is scattered among different tables which are related to each other. An electronic database package designed to manage relational databases is called a relational database management system or RDBMS.

RDBMS packages are able to handle various levels of complexity. Desktop databases (Microsoft Access, Microsoft FoxPro, Borland Paradox) are among the simplest packages. They provide enough power for home or even small business use. More powerful packages (Microsoft SQL Server, ORACLE, IBM DB2) are used to manage data in large corporations. They are designed to manage millions of records, and they offer a very large feature set.

RockProp is powered by SQL Server 2000 (developed by Microsoft). A large server-based database was chosen over a smaller desktop package because it is expected that RockProp will eventually host large amounts of data. In addition to inability to effectively manage large data sets, desktop databases are poor at handling multiple simultaneous user connections. Other criteria that influenced the selection of a RDBMS included the ability to run under the MS Windows environment, ease of use, availability of information about the package and its adequate support in the press (books), and easy comprehensive accessibility through the Internet. MS SQL Server was designed for the PC platform (MS Windows based). It can be
easily accessed through the internet using ASP. The SQL Server, being a Microsoft’s product, ensures its smooth integration with other Microsoft’s technologies used: MS Visual Basic and ASP. A significant number of books are available on the package. For these reasons Microsoft’s SQL Server 2000 was chosen for the server side RDBMS in RockProp.

3.2.2 Database Objects: General to All Packages and Specific to SQL Server

This subsection provides some background on modern relational database technology, database design techniques, and some related vocabulary.

A relational database consists of tables. This is true for any RDBMS. There are several reasons for information to be subdivided among tables. A single-table (flat) database has a number of disadvantages: data redundancy; data deletion, insertion, and update anomalies. In order to demonstrate these, imagine there is a database for books in a library. When several records (books) in the database share the same publisher, it is preferable to put information about books and publishers into separate tables and join them rather than to have all information in one table. First, this allows a publisher record to be entered once in the publisher’s table. The publisher’s information is then linked to every book that it published, and is listed in the books’ table. This eliminates the need to enter full details of a publisher for each book. Having complete publisher information repeated with every book creates data redundancy. The need for this information to be entered with every book record often results in an insertion anomaly. Separation of information into two tables eliminates both of these (Roman, 1997, p.6,7).

Having both book and publisher information in one table would also caused other anomalies. When a publisher’s information, for example address, changes, it has to be changed in every
book entry. This is an *update anomaly*. A separate table for publishers allows change of information to occur in one place only. It is possible for a database to have only one book published by a particular publisher. If the book is lost, its record has to be removed from the database. Deletion of the record in this case removes the publisher’s information from the database as well. This is called a *deletion anomaly*. Publishers are independent of the books they publish; keeping their information in a separate table avoids information loss (Roman, 1997, p.6,7).

When data is split among several tables, there has to be a mechanism to keep track of which rows in one table correspond to rows in another table. This is accomplished through the use of primary keys, foreign keys, and relationships.

Every table in a database usually contains a primary key (PK). A *primary key* is a table column (also called a table *attribute*) or a combination of columns that uniquely identifies every row in the table. When two tables are joined together, every row of one table must contain a pointer to some row in another table. This pointer is called a *foreign key* (FK). FKs generally point to PKs since those are unique. A *relationship* is a database object that defines which FK is related to a PK and the rules of such correspondence. Relationships, primary and foreign keys are not dependent on RDBMSs.

Besides general (common to other RDBMSs) database objects, there are many other objects that are application-specific. A few of these are briefly described below.

Stored procedures and views are two of the most frequently used SQL Server objects. A *stored procedure* is a database object programmatically defined by set of statements and stored in the
database. Stored procedures can be executed within a database or remotely by a database user. Stored procedures can accept and return values; they can be considered as a database’s little computer programs. A view is a virtual table. Unlike a normal table, views do not contain real data; instead, they contain computer code that assembles tables on demand. The purpose of a view is to control what its users are or are not allowed to see (Vieira, 1999, p.34, 35).

Other SQL Server objects that will be referred to in the future are logins, users, and roles. These objects are complimentary to each other. A login allows a person to get access to the SQL Server RDBMS. A user in a database is a login that was granted access to the database. A role defines privileges (limitations) of a user. Database users are assigned to particular roles.

When a new database is created in SQL Server, several objects (tables, stored procedures, logins, roles, etc.) are automatically generated by RDBMS. These objects are used internally by RDBMS; they have little to do with how databases are developed. This chapter presents only user-defined objects. SQL Server specific objects will only be mentioned in cases where they were used during the development process.

### 3.2.3 Main Database: RockProp

*RockProp* is the main database of the project. It contains data related to intact rock properties. This section will introduce its general structure, table layout, and present other information relevant to the understanding of *RockProp*. 
3.2.3.1 Structure

The structure of the RockProp database is similar to the data hierarchy presented in Chapter 2 (Figure 2.1). The database is a collection of records. The record is a single database entry; every piece of information in the database belongs to a record and is governed by it. For the purpose of database use, the record is considered to be a single piece of related information that provides a set of values describing one or several intact rock properties. Each record contains one or multiple sub-records. Every sub-record contains information relevant to a particular property of a record. A record also contains information about its data source, reference information.

3.2.3.2 Table Layout, Primary and Foreign Keys, Relationships

The RockProp database contains 29 tables. Table names, their columns, and relationships between them are shown in Figures B.1-B.4 (Appendix B). Appendix B also contains complete documentation regarding every table, stored procedure, view, and role in the RockProp database.

The physical layout of tables in the RockProp database (Figure B.1) can be subdivided into 3 parts: the record (main) section, the sub-record section, and the reference section (Figures B.2, B.3, and B.4 respectively). Two tables, tblSubRecordSet and tblReferences, are shared between the main section and their corresponding sections, sub-record and reference sections, respectively.
Main Section

The main section of the RockProp database (Figure B.2) contains record information of descriptive nature: material origin and its description, classification characteristics of sub-records. The governing table of the main section, as well as of the entire database, is `tblRecordSet`. Every record in the database is uniquely identified by an integer number which is generated by the database when it creates a new record. This is the `RecordID` attribute of the `tblRecordSet` table which is also its PK. Each record’s spatial origin is described by the pair of geographical coordinates (latitude and longitude) and depth below ground surface.

More comprehensive location of every record is given by country and geographical region. Every country in the world is listed in the `tblCountries` table to which `tblRecordSet` is linked through the `CountryCode` attribute (FK in `tblRecordSet`, Figure B.2). Country names were put into a separate table to minimize data redundancy. Region names are listed in the `tblRegions` table to which `tblCountries` is linked through the `RegionCode` attribute (FK in `tblCountries`, Figure B.2). Again, in order to minimize the data redundancy, a separate table was established for geographical region names. This country and region referencing was implemented to facilitate information searches by geographical regions and countries.

Every record has a description of the rock material. `tblMaterials` is the table that was designed to hold this information. It is linked to `tblFabrics` and `tblRockTypes` through FKs (FabricID and RockTypeID respectively in `tblMaterials`, Figure B.2). Rock type and fabric names were put into separate tables to avoid data redundancy. The main material table (`tblMaterials`) is linked to the main database table (`tblRecordSet`) through the `RecordID` attribute that appears in both tables.
Every sub-record in the database is held in `tblSubRecordSet` table (Figure B.2). Each sub-record has a unique ID (`SubRecordID` attribute, PK) which is generated automatically by the database when a new sub-record is created. It was mentioned before that each sub-record is a component of a record. In order to correlate sub-records to a corresponding record, `tblSubRecordSet` contains the FK, `RecordID`, which links the table to `tblRecordSet`. This mechanism allows for the correspondence between the two. Every time a new sub-record is created, the database inserts its parent’s `RecordID` into the appropriate column. It is possible to input the quality of a sub-record information; the `InformationQuality` attribute of the `tblSubRecordSet` table holds integer values from 1 to 5. The value of “5” is given to a high quality (reliable source) data; the value of “1” is given to a feasible (not from a reliable source) data. This allows the user to check on the data source reliability.

Two additional tables, `tblDataTypes` and `tblDataSources`, are referenced from `tblSubRecordSet` table with `DataTypeID` and `DataSourceID` FKs (Figure B.2). These tables contain sub-record classification parameters. As described in Chapter 2, `data type` classifies a sub-record based on the testing category type; at present the `intact` data is of primary concern. Another possible category is `in-situ` data. The `data source` classification is used for determination of test types that data was determined from. This will be described in more detail when the sub-record section is presented. `tblDataSourceGroups` contains groups for the data sources which are used to classify them on the basis of general parameter meaning. These groups include strength parameters, index properties, etc. This table is referenced from `tblDataSources` table using the `GroupID` attribute (Figure B.2).
The last table shown in the main section of the *RockProp* database’s table layout is
*tblReferences* (Figure B.2). This table is actually a part of the reference section. It is shown in the main section to demonstrate how it relates to the rest of the data. More information about this table is given in the reference section.

*Sub-Record Section*

The sub-record section of the *RockProp* database table layout (Figure B.3) is where tables that store actual geotechnical data parameters are located. The structure of this section is fairly simple. It contains two types of tables (excluding *tblSubRecordSet* table): 8 tables contain actual data and 4 tables contain geotechnical test names related to a particular testing method.

Tables that contain actual data (*tblSlakeDurabilityTestData*, *tblIntactHoekParameterData*, *tblIntactVelocityTestData*, *tblSchmidtHammerTestData*, *tblIntactCompressionalTestData*, *tblIntactTensileTestData*, *tblIntactFractureToughnessTestData*, and *tblIntactPointLoadTestData*) all share a common parameter, *SubRecordID*. This parameter determines which sub-record the data in every table belongs to. Besides being a PK for every data table, the *SubRecordID* is also an FK; it correlates entries to appropriate rows in the *tblSubRecordSet* table. When a new sub-record is created, the database determines which table to insert data into, based on selected data source for the sub-record. The database also ensures that one, and only one, of the eight tables contains a particular *SubRecordID*. Data for any given sub-record can only be of one data source and stored in one table.

The remaining four tables (*tblIntactCompressionalTestTypes*, *tblIntactTensileTestTypes*, *tblIntactFractureToughnessTestTypes*, and *tblIntactPointLoadTestTypes*) are used to store
particular test types and are referenced from appropriate data tables (Figure B.3). This eliminates data redundancy.

Reference Section

The reference section contains tables used to store data reference information. The main table, \textit{tblReferences}, is referenced by \textit{tblRecordSet} using the \textit{ReferenceID} FK. This allows several records to relate to the same reference.

The main reference table contains information common to any type of reference: \textit{Title}. Every available reference type is listed in the \textit{tblReferenceTypes} table that is linked from \textit{tblReferences} using the \textit{ReferenceTypeID} FK.

The concept of having different tables for different reference data types is exactly the same as the one described in the previous section, relating sub-record data tables to a main sub-record table. Reference tables store the \textit{ReferenceID} which determines the title of the reference and which record it describes.

The last and the most interesting feature of this section is how authors’ details are implemented. Authors’ last names and initials are stored in the \textit{tblAuthors} table, the PK of which is \textit{AuthorID}. Many of the references have multiple authors, and the same person can be an author of many references. Such a relationship was implemented using an additional table, \textit{tblReferenceToAuthor}, that relates \textit{ReferenceIDs} to appropriate \textit{AuthorIDs}. Such implementation elegantly solves the multiple author problem.
3.2.3.3 Other Database Objects: Stored Procedures, Views, and Roles

Besides tables, the *RockProp* database contains a number of stored procedures, views, and one specific role. Each of these objects has little relevance to the database design; all of them were introduced during the application and the middle tier design stage. Their description, therefore, is postponed until an appropriate section.

Complete documentation on properties of tables, stored procedures, views, and roles of the *RockProp* database can be found in Appendix B.

3.2.4 Satellite Database One: GeoRef

*GeoRef* is one of the project’s two additional databases. It contains geographical coordinates (latitude and longitude) of many world geographical features, construction projects, and populated places. The *GeoRef* database emerged from the idea of being able to search for geotechnical data based on the location of a nearby town, village, dam, mine, etc.

The idea came from Dr. Curran (author’s supervisor) and Dr. Corkum (Rocscience Inc., a geotechnical software developing company). They suggested that it would be convenient and natural for future users to search for data around major cities by providing the radius of search. The author developed this idea, taking it a little further; besides cities, a number of other features like mines, dams, water falls, etc. were incorporated. Table C.1 (Appendix C) provides a complete list of features referenced in the database.

Finding appropriate data for this database was the most challenging part. Data was found on the Internet. The National Imagery and Mapping Agency (NIMA), in the United States, provides
data files for the majority of the countries free of charge (NIMA [1], 2000). Country codes used in the database and interpretation of the feature codes were found on the NIMA’s web server as well (NIMA [2], 2000; and NIMA [3], 2000). Names of the first order administrative divisions (states in the US, provinces in Canada, etc.) were also obtained from NIMA (NIMA [4], 2000). NIMA does not provide information for the US and Antarctica. Similar information about the United States and Antarctica, however, was obtained from the United States Geological Survey web server (USGS, 2000). Regions that the countries were assigned to were based on each country’s location.

The GeoRef database is designed to adopt the information obtained from NIMA and USGS. Tables C.1 – C.3 (Appendix C) list features used for the database purpose (the rest were filtered out), geographical regions (including their codes), and countries (with codes). A great deal of effort was invested into transitioning of the data from data files into the database, filtering of unwanted features, and correlation to regions.

The remainder of this section describes the general structure of the GeoRef database, its table layout, and presents other relevant information. The database use will, however, be discussed when the middle tier and the application are described.

3.2.4.1 Structure: Table Layout, Primary and Foreign Keys, Relationships

The structure of the GeoRef database is fairly simple (Figure C.1, Appendix C). It contains 11 data tables and 4 accessory tables. Figure C.1 shows only 5 tables (1 data and 4 accessory) in order not to unnecessarily complicate the picture; the remaining 10 data tables have an identical structure to the one shown. Reasons for this are discussed below.
Data obtained from NIMA and USGS was enormous: 3,424,926 records after filtering. It was anticipated that searches for particular features would take a long time and considerable computer resources. To reduce the search time, 11 identical data tables, one table for each geographical region, were created in the database instead of a single data table. These tables are holders for the data. Their names are listed in Table C.2. A detailed description of the GeoRef’s structure will be done using a single table (tblDataAF).

Just as in the RockProp database, the RecordID attribute is used as the PK in each data table. The difference is that it is not unique in the entire database. Numerous identical RecordIDs appear across data tables. Since data is located in different tables that are not related among each other, it is acceptable. Every data table contains latitude, longitude, feature’s full name and its characteristics.

The feature characteristics (feature type, country and administrative division (ADM) of origin) are contained in separate tables: tblFeatureType, tblCountries, and tblADM1. They are referenced from the data tables (tblDataXX) using appropriate FKs: FeatureCode, CountryCode, and ADM1, respectively.

Table tblCountries contains a special attribute Present. This binary column is used to indicate which country’s data is referenced in the database. The use of this feature is discussed in the middle tier and the application sections. This table also references the tblRegionCodes table that contains names of the geographical regions, using the RegionCode attribute.
3.2.4.2 Other Database Objects: Stored Procedures, Views, and Roles

The GeoRef database contains a number of stored procedures, views, and one specific role. Each of those objects has little relevance to the database design; all of them were introduced during the application and the middle tier design stage. Description of these objects is postponed until the appropriate section.

Complete documentation on the properties of tables, stored procedures, views, and roles of the GeoRef database is provided in Appendix C.

3.2.5 Satellite Database Two: RPUsers

The RPUsers (short for RockProp Users) database is the second additional database in the project. It was designed to keep track of RockProp’s users.

3.2.5.1 Structure: Table Layout, Primary and Foreign Keys, Relationships

The RPUsers database consists of 4 tables (Figure D.1, Appendix D). Its main table is tblUser; it serves as a container for users’ details. The UserID and UserPass table attributes contain login name and password that allow users to access the RockProp database. UserID is unique for the table and therefore is its primary key. The table is designed to store some additional user information: full name (Name), email address (Email), company name (Company), and home country. Country names are stored in the tblCountry table which is referenced from the main table using the CountryID FK.

Besides information supplied by the user during registration, the main table contains information about the users accessing of the database. When a user registers, the database
automatically inserts the date and time of the registration into the \textit{RegistrationDate} column. The RDBMS inserts the date and time of access instance every time a user logs-in to use \textit{RockProp}. This information is stored in the \textit{LastLoginDate} column of the table. The \textit{AccessTimes} column stores data about how many times a user logged-in into \textit{RockProp}. This column contains an integer value that is incremented by one every time a user is granted access.

The main table uses the \textit{UserTypeID} attribute to reference the \textit{tblUserType} table. Each registered user has certain privileges. A majority of users can only view data stored in the \textit{RockProp} database. They are designated with \textit{GST} value in the \textit{UserTypeID} column) indicating they are guests. Users that are allowed to add data to the database and to view user statistics have \textit{ADM} (administrator) value in the \textit{UserTypeID} column. Currently, the only person with these privileges is the author.

The last table in the database is the \textit{tblAccessInstance} table (Figure D.1). This table stores information about every database access instance by any user. Every time a user logs into the database, the user’s login name (\textit{UserID}) and date of access (\textit{AccessDate}) are written into this table. The user machine’s IP address (\textit{IPAddress}), name of the server the connection originated from (\textit{ServerName}), and web browser type (\textit{HTTPUserAgent}) are also stored in the table.

\subsection*{3.2.5.2 Other Database Objects: Stored Procedures, Views, and Roles}

The \textit{RPUsers} database contains several stored procedures and views. Each of them was created during the application and the middle tier design stage. In order to understand their purposes, it is best to describe them in the appropriate section.
Complete documentation on properties of tables, stored procedures, views, and roles of the
RPUsers database can be located in Appendix D.

3.3 **Client and Middle Tiers**

The web application is the only client of the RockProp RDBMS at present. The middle tier was
developed specifically to suite web application needs. The description and discussion of both
the client and the middle tiers should be presented together in order to understand how they
function. This section presents general information about both tiers: description, interaction
with users, data handling, database access scenario, and RockProp’s physical setting. It is best
to provide a description of actual functions used to access the database along with the
functionality of the client tier (Chapter 4).

3.3.1 **Client Tier**

Being part of the 3-tier architecture, RockProp’s web client is responsible for the presentation of
information. A web browser is used by the user to access data. When the Internet was first
created, web browsers could only interpret web pages written in HTML (Hypertext Markup
Language). HTML web pages are static; their content is predefined by developers. Modern
browsers have become more sophisticated; small programs can be attached to or embedded in
web pages. Even though HTML rendering still remains their primary purpose.

Database information can not be delivered using pure HTML. It is impossible to predict which
information users will want to access and make an appropriate web page for it. Moreover, such
predefined pages quickly become obsolete when new information is added to the database. This problem is resolved by using dynamic web pages.

*RockProp*’s client tier is divided between the user’s web browser and the server-side dynamic web page generation implemented using Microsoft’s Active Server Pages.

### 3.3.1.1 ASP Technology

Active Server Pages (ASP) is a Microsoft technology for generation of dynamic and interactive web content. ASP technology consists of a scripting engine located on the web server. ASP web pages contain standard HTML instructions as well as server-side scripting code. When a user requests the page, the ASP engine searches for the embedded code, executes it, and returns results. Any HTML included within the page is simply output without change. The result of the ASP page processing is standard HTML that is sent back to the web browser (Denault *et al.*, 2000, p.11-15).

HTML allows users to submit data to a web server. Standard HTML forms are used to accept the user’s input. *Hypertext Transfer Protocol* (HTTP) is responsible for handling communication between the server and the browser. Data submitted by users is embedded into this protocol by the browser and sent to the web server. ASP technology allows this data to be extracted from the incoming HTTP requests. The ASP engine (through embedded code) handles incoming data and determines what to do with it and how to respond. The ASP server side code can be written in either *VBScript* or *JScript* languages. Both of them are functionally equivalent; the developer chooses which language to use. Every ASP page for *RockProp* was developed using *VBScript*. 
Being executed on the server and having extended programming functionality, ASP technology allows easy access to the middle tier. ASP’s ability to process user input (requests) and to communicate with the middle tier allows RockProp users to access database information. In RockProp, the ASP engine is responsible for generating HTML output that users see in their browsers.

3.3.1.2 Web Browser

The web browser is responsible for the HTML rendering and presentation of content on the client’s machine. Most modern browsers on different computer platforms can be used to access RockProp. Some old browsers and new browsers with disabled functionality, however, can not use RockProp. Several requirements must be satisfied by web browsers in order for them to use RockProp.

A web browser must support HTML frames. Numerous ASP pages were created using frames. This feature is supported by all current browsers and should not be a problem for a majority of users.

In order to access the RockProp database through the Internet, a web browser must support JavaScript 1.2 which must be enabled in the browser preferences. JavaScript is an object-based, cross-platform computer language developed by Netscape and Sun Microsystems. Being implemented in a majority of today’s browsers, it allows web developers to embed client-side routines (code) into HTML pages. JavaScript, running inside the web browser, adds much functionality to static HTML web pages without the need for server-side processing (Wyke et al., 1999, p.3-5). In RockProp, JavaScript procedures are responsible for several input
validating and submission functions, presentation functionality, etc. They help to improve the quality of the presentation, reduce the number of unnecessary calls to the web server, and reduce general web server load. Chapter 4 provides more details on this subject.

The application was tested with both Microsoft Internet Explorer and Netscape Navigator. Although it works without a problem with the Netscape browser, RockProp is best accessed using Internet Explorer version 4 or higher. Internet Explorer can be downloaded free of charge from Microsoft Corporation’s web site at http://www.microsoft.com/windows/ie/default.asp.

3.3.2 Middle Tier

The middle tier of RockProp is implemented as a combination of database access functions in an ActiveX DLL library and a set of stored procedures and views within each of the three databases. An ActiveX DLL library is a specially compiled binary file. Its special form allows it to be used from the ASP engine. The library file was developed using Microsoft’s Visual Basic 6, an object-oriented programming language. The file contains numerous classes (sets of functions) designed to be executed from the ASP web pages. Many of them perform specific database tasks (user registration, data searches, etc.) by calling stored procedures and views implemented in the databases. Both stored procedures and views are written in Structured Query Language (SQL). SQL (pronounced SEQUEL) is the most popular commercial programming language developed for relational database systems; its main distinction from other languages is that “users specify what data they want, not how to get it” (Bonner, 2001, p.1-6).
Implementation of data access functionality using a RDBMS’s stored procedures and views, rather than doing everything in the middle tier DLL library, significantly simplifies maintenance of the middle tier. The main reason for this due to the way both components are updated. Stored procedures and views are database objects. They are stored in the form they are written. They are easily updated by the database administrator. The middle tier DLL library file, however, is compiled. The binary format (compiled form) can not be changed (modified) without file re-compilation. This can become a big maintenance drawback. In *RockProp*, a majority of data accessing is done using stored procedures and views which can be easily modified.

### 3.3.3 Interaction Mechanisms Between Web Application and Middle Tier and Between Middle Tier and Database Server

The interaction between the browser and the ASP engine was briefly mentioned before. The web browser uses HTTP protocol to retrieve an ASP web page. The web server passes request for the page to the ASP engine. The ASP engine executes code in the page, obtains results, and passes composed HTML back to the web server, which sends it back to the browser via HTTP.

Interaction between the ASP engine and the middle tier DLL library is straightforward as well. When an ASP page needs to use the middle tier, it calls an appropriate function from it and submits the required parameters. When the middle tier finishes processing, it passes results back to the ASP engine which picks up page processing where it last left it and continues execution of the ASP code.
Interaction between the middle tier and the database server is more complex. The middle tier cannot access the RDBMS directly. It uses a data provider, which is the interface through which the middle tier communicates with the RDBMS. The RockProp's middle tier DLL library uses ActiveX Data Objects (ADO) to interact with the SQL Server. ADO consists of a number of objects that expose different aspects of SQL Server functionality to the middle tier. ADO makes it possible for the middle tier to operate: connect to the database, execute stored procedures, retrieve data, etc.

This is a brief overview of the communication mechanism. Interested users should refer to Chapter 4 which describes functions, stored procedures, and views developed for data access.

### 3.4 Physical Location of Components

Several physical configurations are possible for RockProp. The simplest configuration is to have the web server with the ASP engine, the middle tier and the RDBMS on a single machine. This organization allows easy communication between the components. This is also the best approach during the development process. Problems, however, might start arising when a substantial number of users start accessing the server simultaneously. In this situation users would start to experience delays both in obtaining data from the database and getting access to the web server.

One possible solution to this problem is to put the web server with the middle tier on one machine and the database server on another, linking them through a network. In this case, more computational resources will become available to both the web server and the RDBMS server. While providing a good solution, this approach has its own complications. The network
connection may become the bottleneck in the setup. If users request a large amount of data, then the network may become clogged with data transmission, causing the equivalent delays.

The issue of component location is highly dependent on the number of users visiting the database, quantity of data in the database, and amount of data retrieved by users. All of these require processing power and affect the network traffic. This issue should be investigated in the future once the server load is known. At present, RockProp is in the development stage. It is currently situated on a single machine which has, so far, proven to be sufficient for its effective operation.
CHAPTER FOUR – ROCKPROP’S FUNCTIONALITY AND ITS IMPLEMENTATION

This chapter describes the functionality of the RockProp’s web interface. First, it illustrates features available to Internet users. Second, it provides implementation details of these features with an emphasis on the technical aspects. The last section of this chapter briefly describes requirements, installation and configuration of the RockProp application.

4.1 THE ROCKPROP WEB INTERFACE FUNCTIONALITY

The functionality of the web interface can be divided into 3 parts: user registration and login, data searching, and data presentation. These features can be accessed by all RockProp users. Any user using RockProp can only access it through the main page (Figure E.1, Appendix E). This page lists browser requirements and indicates if the necessary functionality is enabled in the browser. It also contains two buttons (login and register) which allow new users to register and registered users to login into the database.

4.1.1 User Registration and Login

Any person is authorized to access data in the RockProp database through the Internet. Before access can be granted, a user must register. Registration is a simple process and provides immediate access to the data.
The registration web page can be opened by pressing the register button on the main page (Figure E.1). Registration information is entered in a new window (Figure E.2) that is opened by pressing the register button. In order to register, a user must provide his/her full name, an email address, a login name, and a password. Optional registration parameters include company and country. The registration process is completed by pressing the submit button and receiving a confirmation message. Registration will be declined only if the submitted login name is already in use by someone else. In this case, the user must choose a different login name.

In order to access the data in the database, users must first login into RockProp. An appropriate dialog can be opened by pressing the login button on the main page. Figure E.3 shows the login window. To login, users must provide the login name and the password they entered during the registration. After a successful verification of the supplied information, users get access to the main application window which is opened automatically (Figure E.4). This window provides various data search options, as shown on the left side of Figure E.4.

4.1.2 Data Searching

There are five data search methods implemented at present: rock type search, search by geo-coordinates, search by geo-region and country, and geo-feature search methods. Users are asked to enter the parameters necessary for a selected search method. The results of a search method are records that match specified search criteria; they are displayed in a standard format after a search procedure is complete.
4.1.2.1 Searching by Rock Type

This search method extracts every record in the database wherein rock type matches the entered value. The rock type search is activated by pressing the by rock type link in the main window menu (Figure E.4). This loads an appropriate search page to the right of the menu (Figure E.5 (a)). The only parameter required for this search method is rock name which is entered in the text box. Figure E.5 (a) shows an example of the database search for granite. The results of this search example are presented in Figure E.5 (b). Record information can be viewed by activating the rock name link or by pressing on the image below the record number (Figure E.5 (b)).

4.1.2.2 Searching by Geographical Coordinates

This search method extracts every record within the specified search radius around a particular point, designated by the geographical coordinates. It is initiated by pressing the by geo-coordinates link in the menu section of the main window. Figure E.6 (a) shows a web page used for entering the search parameters. In order to search the database by geo-coordinates, users must supply latitude and longitude values in decimal degrees. These coordinates designate the center of the search. The latitude values must lie between -90 degrees (south of the equator) and 90 degrees (north of the equator). The longitude parameter field will accept values between -180 degrees (west of the Greenwich meridian) and 180 degrees (east of the Greenwich meridian). The radius of search in kilometres is entered in the search radius text box. Figures E.6 (a) and E.6 (b) provide an example of the results obtained with this search method.
4.1.2.3 Searching by Geographical Region and Country

Searching by geographical region and by country are related. They were designed to obtain all data for a particular geographic region (searching by geo-region) or a particular country (searching by country). Each method is activated by its own link in the menu section of the main application window. Figures E.7 (a, b) and E.8 (a-c) show the parameter selection and results of the geo-region and country search methods.

The name of a region is the only parameter required for the geo-region search (Figure E.7 (a)). The name is selected from the dropdown box in the web page. The dropdown box only lists names for regions that are referenced by the database data.

In order to search the database by country, users must specify the region first (Figure E.8 (a)). Upon proceeding to the next page (Figure E.8 (b)), users must select a country for the search. The dropdown box only lists countries that belong to the specified region and which are referenced by the database data.

Both search methods produce the standard search result page (Figures E.7 (b) and E.8 (c)). Record information can be viewed by activating the rock name link or pressing on the image below the record number.

4.1.2.4 Searching by Geographical Feature

The geo-feature method was designed to search for records around a particular geographic or construction feature, as well as around cities, towns, and villages. This search method is initiated by pressing the by geo-feature link in the menu section of the main window.
In order to execute the search, users must specify a number of parameters related to a feature’s location. These are presented in Figure E.9 (a-e). First, the region of the feature’s location must be selected. This is selected from the dropdown box which lists every region specified in the GeoRef database (Figure E.9 (a)). After the next button is clicked, the method requires the user to enter a country of interest. Countries located in the selected region are listed in the dropdown box (Figure E.9 (b)). If the user does not know the name of a country, he/she is allowed to choose the any value; this will perform the search within the entire selected region. Next, the user is asked to select the name of the first-order administrative division. The dropdown box used to make the selection only lists values applicable to the previously selected country (Figure E.9 (c)). If no country is selected (search is performed over the region), then this dialog is omitted; the user will be automatically transferred to the next selection window. The next step requires the user to choose the type of feature and enter the feature’s name and search radius. The feature types listed in Table C.1 (Appendix C) are taken from the GeoRef database. The feature name is an optional entry; the radial selection buttons allow searching by either specifying the name of the feature or by the feature type alone. The search radius text box accepts a value (in kilometres) for the radius, around the feature, within which the search should be executed (Figure E.9 (d)).

The search is executed in two stages. First, the user is presented with a list of features that match the selected criteria (Figure E.9 (e)). This allows the user to choose an appropriate feature around which the actual record search will be executed. This is helpful when no, or only a part of the, feature name was supplied. If the feature search returns more than 100 instances, the user is asked to specify additional parameters to narrow the search. The second stage in the
feature search is executed manually; a particular feature must be selected by clicking on its name in order for the search to continue. This results in a standard record selection screen (Figure E.9 (f)) from which record information can be viewed.

4.1.3 Record Information Viewing

Complete information for any record can be retrieved by activating the rock type link or pressing on the image below the record number of the standard search result window. (Figures E.5 (b), E.6 (b), E.7 (b), E.8 (c), or E.9 (f)). This operation opens a new window in which information is displayed (Figure E.10 (a)). Figure E.10 (a-e) provides an example of how information is presented.

The top of the information window displays the record number within the ones selected by the search. The window also contains a standard set of navigation buttons that allow easy access to other records found during the search. The prev and next buttons display information about the records just before and after that produced by a search (Figure E.10 (a)). The first and last buttons display information about the first and last records of all the records selected by a search (Figure E.10 (a)).

When information about a record is first displayed, a user can observe the record’s general information (Figure E.10 (a)) which includes rock type, location information, and some material properties. This is the default view; whenever a user navigates through the record-set using the buttons, general information is always displayed first. Other available information can be viewed by selecting appropriate tabs: intact data tab or reference tab.
The **intact data** section contains a record’s intact data parameters. This section also contains several subgroups that are based on sub-record groups discussed in Chapter 3. The record shown in Figure E.10 (a-e) contains information on 3 subcategories of **intact data**. Only categories that are referenced by a record are listed under the **intact data** section. If a record does not have information, about **index properties** for example, this subcategory will not appear.

In order to view data from different subcategories, users must click on a subcategory name. Figure E.10 (b-d) shows how information is displayed.

The last section in the information window (**reference**) contains a record’s reference information (Figure E.10 (e)). It contains a complete reference to the source of the data.

### 4.2 Implementation of Functionality: Technical Description

This section describes technical details of RockProp’s implementation which is closely related to the functionality described in the previous section. The description includes ASP pages, their interaction with the middle tier, middle tier functionality, and database access. In order to make this section easier, the material is presented starting with the origin of a request, ASP pages. It then covers how the middle tier responds to the request and how the ASP engine supplies information to the browser. Figure 4.1 shows the application’s directory structure.

![Figure 4.1 Web application’s folder structure.](image-url)
4.2.1 User Registration and Login

The default page for the project is *default.asp* which is located in the root directory. This page is shown on Figure E.1. The ASP code for the page can be found in Appendix F. This page has several important roles. First, it provides information about *RockProp*, lists system requirements, and displays information about the user’s browser. *RockProp* information and system requirements (shown on the right side of the page) are contained in the *welcome.txt* file located in the same directory. Putting this information into a separate text file allows it to be easily edited without being lost within the scripting code. The content of the *welcome.txt* is automatically inserted into the web page by the ASP `server.execute` command which reads the file content and inserts it into the output. The user’s browser information is displayed on the left side of the page. This is accomplished by using several built-in JavaScript functions which, running inside the web browser, obtain browser information when the page is displayed.

The *default.asp* page also allows the user to open the registration and login forms (Figures E.2 and E.3). The registration window is opened when users click on the *register* button. This activates a JavaScript function (*showRegForm*) within the *default.asp* page. This function opens a new browser window and loads the *newuser.asp* page, located in the *newuser* folder. The login window opens when users press on the *login* button; this opens a new window containing the *login.asp* file which is located in the root directory. This is accomplished by activating a JavaScript function (*showLoginForm*) within the *default.asp* page.
4.2.1.1 User Registration

The user registration process starts when a new window with the newuser.asp file is opened. The contents of this file can be found in Appendix F. The newuser.asp shows an HTML form with predefined text fields for user information input. Use of text fields is straightforward. The form additionally contains a drop down list box with country names.

Obtaining Country Names

Country names are obtained by using the GetAllCountries function of the clsUser class of the middle tier. The printout of the clsUser class is found in Appendix G. When an instance of this class is created, a connection to the RPUsers database is made. The GetAllCountries function uses the vwAllCountries view (Appendix D) in the database to get all country names. The view is used for easy maintenance; it can be changed inside the database without affecting the middle tier function. The GetAllCountries function returns a 2-D array with country names to the newuser.asp page.

Before registration information is submitted to the processing page, the newuser.asp performs checking of the input. A set of JavaScript functions ensures that users submit the required information. This client-side error checking saves a round trip to the web server in case of an error.

When users submit registration information, the createnewuser.asp file, located in the same directory, becomes the recipient of the information (Appendix F). Submitted information is extracted from the HTTP protocol and stored in local variables. The createnewuser.asp file
then executes the AddNew function of the clsUser class (Appendix G) in the middle tier, supplying the required information to the function.

Registration of a New User

The AddNew function in the middle tier library file is responsible for registration of new users. When an instance of the clsUser class is created, it sets all necessary parameters for the connection to the RPUsers database. The AddNew function opens the database connection and executes spAddNewUser stored procedure (Appendix D) in the database, supplying necessary parameters to it. The spAddNewUser stored procedure first checks if the database already contains a user with the supplied login name. If the login name is in use, the stored procedure aborts and returns the value of 2 (two). If no user with such a name is found, the spAddNewUser creates a new user and returns the value of 0 (zero) indicating to the middle tier that the operation was successful.

When the AddNew function of the clsUser class receives the return code from the executed stored procedure, it first closes the database connection. Then it checks the return value supplied by the stored procedure; if the value is not zero, an error flag is raised with an appropriate message. Disregarding the stored procedure output, the AddNew function returns control back to the createnewuser.asp page.

After the AddNew function is executed, the createnewuser.asp page checks if any errors were raised. If no errors are present, the page displays the registration confirmation message. If an error is present after the execution, the execution of the page is aborted, the error message is
extracted from the error objects, and control is transferred to the errhandler.asp page in the root directory.

**Error Handling**

The error handling ASP page (errhandler.asp, Appendix F) is designed to output error messages. Every time it is called, it reads an error description from the ErrDescription session property. It then outputs it to the screen, providing an opportunity for the user to go to a previous page and correct mistakes.

4.2.1.2 User Login

User login is performed using the login.asp page located in the root directory (Appendix F). This page displays an HTML form which accepts the user’s login name and password. This form submits data to the loginresponse.asp page. The loginresponse.asp page (Appendix F) processes the user’s input and executes the LoginIntoRockProp function of the clsUser class, submitting the user’s login name, password, IP address, user’s server name, and HTTP agent information to it.

The LoginIntoRockProp function of the clsUser class logs the user into the RockProp database. When called, it executes the spLoginUser stored procedure in the RPUsers database, submitting the necessary information to it. The spLoginUsers stored procedure first checks if a user with the submitted login name and password exists. If it does, the stored procedure retrieves the user type (determines privileges); it then returns the user type value to the middle tier as well as a return code of 0 indicating that the operation was successful. If such a user does not exist, the stored procedure returns a code of 2.
The middle tier checks the return code immediately after the execution of the stored procedure. If the operation was successful, the `LoginIntoRockProp` function sets the `UserTypeID` property of the `clsUser` class to the value of the user type received from the function’s execution. In case of an unsuccessful completion of the login process, the function raises an error and aborts.

When the `loginresponse.asp` page receives control, it first checks if the execution of the `LoginIntoRockProp` function was successful. If the login process is unsuccessful, the code calls for the `errhandler.asp` page described above. If the login process is successful, then the page reads the user type property and stores it in the global variable as well as user’s login name and password. After completion, the `loginresponse.asp` page opens a new browser window showing the main application page and then closes itself.

### 4.2.2 Main Application Interface

The main interface of the application is shown in Figure E.4 (Appendix E). It contains the menu that loads the appropriate pages on the right when search links are activated. When the interface is opened in a new window after a successful login, the `main.asp` page is loaded. This page is located in the `app` folder; its contents can be found in Appendix F. The `main.asp` page contains an HTML frameset which splits the browser window into three frames. This allows the menu (left frame) to be displayed at all times as well as to control any required change of headings in the upper frame. The main frame usually contains search and result pages.

Another function of the `main.asp` page is to check whether a user is logged in to the database. This is done to prevent users from calling this page directly, forcing them to go through the login process. This functionality is accomplished by checking the `UserID` and the `UserPass`
session variables which are created after successful login. If these variables exist, the user can proceed to the main application window. If these variables do not exist, the application displays an error message asking the user to login.

4.2.2.1 Menu Implementation

The menu is located in the left portion of the main application window. It is used to access different search methods and any other functionality added in the future. The menu’s main page is the menu.asp file located in the left subdirectory of the app folder. Its ASP code can be located in Appendix F.

The menu implementation is straightforward. The menu structure is described in the menu.xml file, located in the same directory as menu.asp. Its contents can be found in Appendix F. The menu.xml is an XML file designed to portray the structure of the menu. The parent group elements represent major headings in the menu; the child group elements contain the menu’s subheadings. Every parent and child group contains the user types attribute used to determine which headings are displayed depending on user privileges. If an ordinary user (guest) accesses the database, the menu will only display headings that are common to every user. If a user has administrative privileges, additional headings are displayed; they allow access to the database’s administrative functions.

In order to display the XML data in the browser, it has to be transformed into HTML on the server using an XSLT transformation sheet. The menu.xsl is the transformation style sheet that defines how the menu is displayed on the screen. It is located in the style subdirectory of the left
folder; its contents can be found in Appendix F. The transformation is performed on the server in the menu.asp page using the Microsoft MSXML parser.

4.2.3 Database Search Pages

Database searches are implemented using separate ASP pages for each search type. Every search page is displayed to the right of the menu. All search pages are located in the search subdirectory of the app folder in their own directories. Before a database search begins, all search parameters are supplied to the searchresponse.asp page. This page is discussed in the next section (Section 4.2.3).

4.2.3.1 Rock Type Search

The page responsible for collecting input for database searching by rock type is rocktype.asp (Appendix F), located in the rocktype subdirectory of the search folder. This page contains a textbox for the user to enter the rock type. When the user initiates the search, the rocktype.asp page first checks the content of the text box. If it is empty, an error message is generated and the user is asked to check the input data. The rock type box cannot be empty. If the text entry box is not empty, the HTML form submits the user’s input to the searchresponse.asp page.

4.2.3.2 Geo-Coordinate Search

The coordinate.asp page is responsible for collecting user input for geo-coordinate searches (Appendix F). It is located in the coordinate subdirectory of the search folder. This page contains three textboxes for users to input latitude, longitude, and search radius. When the search button is pressed, a JavaScript function checks if input boxes are empty and if latitude
and longitude values are within the appropriate ranges. If these conditions are not satisfied, users are presented with a warning message. Otherwise, the page submits the data to the searchresponse.asp page.

4.2.3.3 Geo-Region Search

The collection of information for the geo-region search method is performed by the georegion.asp page (Appendix F), located in the georegion subdirectory of the search folder. The main feature of this page is retrieval of referenced region names and their placement in a dropdown selection box. When the page is loaded, the ASP engine supplies the user’s login name and password to the clcRockProp class (Appendix G) using its UserID and UserPassword properties. Both the user’s login name and password are obtained from the ASP UserID and UserPass session variables. In order to obtain referenced region names, the georegion.asp page calls the GetAllReferencedRegions function of the clsRockProp class.

Obtaining Referenced Region Names

The GetAllReferencedRegions function of the clsRockProp class (Appendix G) is designed to retrieve referenced regions and return them to the ASP engine. The referenced region is a region which is referenced by at least one record. The function uses the vwAllReferencedRegions view of the RockProp database (Appendix B) to get the referenced region names. After the view is executed, the function returns a 2-D array to the ASP engine.

When control returns to the ASP engine, region names are extracted from the array using a simple loop procedure and are written to the dropdown box. Page data is submitted to the searchresponse.asp page when the user presses the search button.
4.2.3.4 Country Search

The collection of information for the country search method is performed by the `country.asp` page (Appendix F) located in the `country` subdirectory of the `search` folder. There are two main features of this page: retrieval of referenced region names and retrieval of names of referenced counties that correspond to the region. When the page is loaded, the ASP engine supplies the user’s login name and password to the `clcRockProp` class (Appendix G) using its `UserID` and `UserPassword` properties. In order to obtain referenced region names, the `country.asp` page calls the `GetAllReferencedRegions` function of the `clsRockProp` class. The procedure was described in the previous section (Section 4.2.2.4) and is not repeated here. When a user presses the `Next` button after selecting a region of interest, the `country.asp` page reloads. Effectively, it supplies data to itself. When the page runs the second time, it checks if a region has been supplied. If not, then this is the first run of the page. Otherwise, the region dialog is omitted, and the user is asked to choose a referenced country from a pull-down menu. In order to obtain referenced country names, the `country.asp` page calls the `GetAllReferencedCountries` function of the `clsRockProp` class, supplying a region code to it.

Obtaining Referenced Country Names for a Specified Region

The `GetAllReferencedCountries` function of the `clsRockProp` class (Appendix G) is designed to retrieve referenced countries and return them to the ASP engine. The concept of the referenced country is similar to the one of the referenced region. The `GetAllReferencedCountries` function uses the `spRetrieveReferencedCountries` stored procedure of the `RockProp` database (Appendix B) to get the names of the referenced countries. This stored procedure retrieves the names of countries based on the supplied region code; if the region code is not supplied, it returns every
referenced country, disregarding regions. After the stored procedure is executed, the function returns a 2-D array to the ASP engine with country names.

Once the ASP engine receives execution control back, country names are extracted from the array using a simple loop procedure and are placed in the dropdown box. Page data is submitted to the searchresponse.asp page when the user presses the search button.

4.2.3.5 Feature Search

The execution of this search method is performed in two stages. First, based on the user input, the GeoRef database is searched for features matching the user criteria. Second, based on the geographic coordinates of a selected feature and radius of search supplied by the user, the RockProp database is searched for records. The second stage is equivalent to the coordinate search method.

Figure E.9 (a-d) shows that parameters are selected using multiple pages. Users submit parameters in several stages; during each stage they are presented with a separate screen. A single page is used for every stage. The feature.asp page (Appendix F) is used to collect user input, bringing data into itself. This page is located in the feature subdirectory of the search folder. It determines which information parameters have already been selected by the user, and alters the output (HTML sent to the user) accordingly.

Obtaining All Region Names

During the first run of the page, the user is asked to select a region of interest from the dropdown box. In order to get all regions, the ASP page uses the GetAllRegions function of the
clsGeoRef class (Appendix G) in the middle tier. This function uses the vwAllRegions view (Appendix C) of the GeoRef database to extract the necessary data. The middle tier function passes a 2-D array to the ASP engine. The region names are extracted from the array and written to a dropdown box element of a standard HTML form. The user-chosen region name is submitted to the same page for a second run.

*Obtaining Countries Based on a Selected Region*

During the second run of the features.asp page it sets the RegionName property of the clsGeoRef class to the previously selected region value. This property will be used in the middle tier to extract countries that belong to the specified region. In order to obtain country names, the ASP engine executes the GetCountries function from the clsGeoRef class (Appendix G). This function executes the spRetrieveCountries stored procedure (Appendix C), supplying the name of the selected geographical region. The stored procedure returns country names which are then sent to the ASP engine in a 2-D array. The country names are extracted from the array and put into a dropdown box element. The selected country name together with the region name from the previous run are submitted to the feature.asp page for the next run.

It is possible to omit the country selection by choosing the _any_ value from the dropdown list. This will force the application to skip the selection of the ADM.

*Obtaining ADMs Based on Selected Country*

During the third run, the page sets the CountryName property of the clsGeoRef to the previously selected country. This property will be used in the middle tier to extract ADMs that belong to the specified country. In order to obtain the names, the ASP engine executes the
GetADMsForCountry function from the clsGeoRef class (Appendix G). This function executes the spReturnADMsForCountry stored procedure (Appendix C), supplying the name of the selected country. The stored procedure returns ADM names which are then sent to the ASP engine in a 2-D array. The country names are extracted from the array and put into a dropdown box element. The selected ADM name, together with all previously selected information from prior runs, is submitted to the feature.asp page for the next processing stage.

Obtaining Feature Types

During the fourth run, the page must obtain names of all existing features. In order to do this, the ASP engine executes the GetAllFeatures function of the clsGeoRef class (Appendix G). This function uses the vwAllFeatures view in the GeoRef database to obtain all feature names (Appendix C). The middle tier function then passes a 2-D array to the ASP engine with all feature names. They are extracted from the array and put into a dropdown box element.

Remainder of Data Input

The remaining data parameters, necessary for the feature search, include the name of a feature and the radius of search. The feature name parameter is optional; if it is not supplied by a user, the search will be performed on the feature type alone. The search radius parameter is simply entered in the text box.

When the last run of the feature.asp page is executed, the file creates a frameset and re-executes itself in the top frame and executes the feature-result.asp file in the lower frame window, passing to it the entire set of user-supplied parameters. The top frame displays a search criteria summary. The bottom frame displays a list of features that match the specified criteria.
Searching for Features

The feature search is executed in the feature-results.asp page (Appendix F), located in the same directory as feature.asp. When the page is executed, it supplies search parameters to the middle tier, using the properties of the clsGeoRef class. Then the ASP engine executes the SearchForFeature function of the clsGeoRef class (Appendix G). This function executes the spFeatureSearch (Appendix C) stored procedure supplying the search parameters. When execution of the stored procedure finishes, the middle tier determines how many features were returned and sets its RecordCount property to this number. Returned information is saved as an XML string which is sent to the ASP engine.

Displaying Features and Executing the Record Search

When the ASP engine receives control, it first checks how many features were returned by the search. This is done by reading the RecordCount property, set by the middle tier. If this number is 0, the execution of the page stops since no features were found; the page displays an appropriate message. If the number of features returned exceeds 100, the page is aborted, i.e. the search result is meaningless because too many features were found. If neither of these is the case, the execution of the ASP page continues. The returned XML string has to be properly displayed on the user’s screen. The feature.xsl XSLT style sheet is used to transform the returned XML into properly styled HTML. The content of this style sheet can be found in Appendix F. In short, this style sheet reads information about each feature and presents it in a table format. It also sets up links in such a way that users only have to click on the feature’s name to execute the search. Every such click executes the standard search page (searchresponse.asp), transmitting the coordinates and search radius of the selected feature.
4.2.4 RockProp Database Data Searches

It was mentioned before that every search page supplies its requests to the searchresponse.asp page that is located in the search directory. Its code can be found in Appendix F. The page itself does not execute the search. This page only creates a frameset for two pages, the sr-top.asp and the sr-main.asp (search directory), and redirects search parameters into them. The upper frame is used to display the search criteria; the contents of the sr-top.asp can be found in Appendix F.

The page in the lower frame (sr-main.asp, Appendix F) is where the database search is executed. First, the ASP engine determines which search method has been requested. The name of the search method is supplied to the middle tier using the SearchMethod property of the clsRockProp class. Second, the ASP engine executes the Search function of the clsRockProp class (Appendix G), supplying the search parameters to it. Search parameters, of course, depend on the search method used. The rock name is supplied when the rock type search method is executed. Coordinates and radius of search are supplied when the feature or the coordinate search are executed. The Search function receives the region name or country name when the country or region searches are executed, respectively.

When the clsRockProp’s SearchMethod property receives the value, it calls the SetSearchProcName procedure within the same class. This function sets the name of the stored procedure that is used to execute the database search. The spSearchByCountry stored procedure is used to search by country. To search by coordinates, the spSearchByCoordinates is used.

The spSearchByRegion and spSearchByRockType stored procedures are used to search RockProp by region and rock type. Printout of these procedures can be found in Appendix B. When the
The **Search** function is executed, the name of the search procedure is already set. When one of the search procedures is executed, it returns values to the middle tier. The recordset is then sent to the ASP engine as XML.

When the ASP engine gets control, it first checks how many records were returned by the search. This is accomplished by reading the value of the `NumOfRecordsReturned` property of the `clsRockProp` class. If the search has returned 0 records, subsequent execution of the page is aborted, and the user receives an appropriate message. If the number of records returned is greater than 0, the ASP engine uses the `searchresponse.xsl` style sheet to transform data from XML into HTML. This style sheet is located in the `style` subdirectory of the `search` folder. Its contents can be found in Appendix F. The style sheet outputs HTML in a such way that when the user clicks on rock name or image below the record number, the `showInformation` JavaScript function is executed. This function opens a new browser window with the `recdata.asp` page and sends it the record number that was activated and the list of all record numbers that were extracted.

### 4.2.5 Displaying Record Information

When a new window is opened with record information, it loads the `recdata.asp` page. This file is located in the `displaydata` folder. Its code can be found in Appendix F. The `recdata.asp` page creates a fairly complex set of frames (Figure 4.2) and loads the `rd-top.asp` page into Frame 2 and `rd-main.asp` into Frame 3 providing them with the chosen record ID and the set of all record IDs.
Frame 2 (rd-top.asp) contains the recordset navigation buttons. Every time a button is pressed, a JavaScript function updates the content of Frame 1, sending it new HTML code. At the same time it refreshes the rd-main.asp page in Frame 3, supplying it a record id for a new record to be displayed. The rd-main.asp page is responsible for extracting data about the current record and displaying this information in Frames 4, 5, and 6. The contents of this page is shown in Appendix F.

As soon as the rd-main.asp loads in Frame 3, the ASP engine executes two functions from the clsRockProp class of the middle tier, GetGeneralDataAsXMLString and GetReferenceDataAsXMLString. These functions return the record’s general and reference data, respectively in XML format; their codes are provided in Appendix G. The first function

![Diagram of frame structure](image)

**Figure 4.2** Frame structure of the data output window.
executes the \textit{vwGeneralRecordData} view from the \textit{PockProp} database. The second function uses the \textit{spGetReferenceInfoForRecord} stored procedure to get the data (Appendix G). Both functions save their results as XML strings and return them to the ASP engine. The ASP engine uses two XSLT style sheets to transform the XML data into HTML. The \textit{generaldata.xsl} file is used for the transformation of the general information XML string, and the \textit{referencedata.xsl} style sheet is used for the transformation of the XML string for the reference data. Both files are located in the \textit{style} subdirectory of the \textit{displaydata} folder; their contents is provided in Appendix F. Transformed HTML for both reference and general data is saved in JavaScript variables; it is not shown to the user directly.

The next step in page processing is to obtain sub-record information for the specified record. The ASP engine uses the \textit{GetRecordDataTypes} function of the \textit{clsRockProp} class to obtain data types available for a particular record (Appendix G). This function executes the \textit{spGetDataTypesForRecord} stored procedure of the \textit{RockProp} database and returns results to the ASP engine. The ASP engine forms a JavaScript function that outputs the data types into Frame 4 as tabs. The ASP engine also obtains groups that data types belong to. This is done by executing the \textit{GetDataSourceGroups} function (Appendix G) from the \textit{clsRockProp} class. The obtained groups are inserted into a JavaScript function in the \textit{rd-main.asp} page which outputs them into Frame 5, after the user selects a particular group in Frame 4.

The sub-record data is obtained by the ASP engine in the following manner. For every data type, the ASP engine executes the \textit{GetSubRecordData} function (Appendix G). This function uses the \textit{spGetDataForSubRecord} stored procedure of the \textit{RockProp} database to obtain sub-record data. The middle tier returns this data to the ASP engine in XML form. The ASP engine
then uses the subrecorddata.xsl style sheet to transform XML into HTML. This file is located in the style subdirectory of the displaydata folder. The contents of the style sheet is provided in Appendix F.

When the ASP engine finishes execution of the rd-main.asp file, it sends the page to Frame 3 of the output window. Upon loading, the JavaScript functions output appropriate data into Frame 6 as users click on tabs in Frame 4 and links in Frame 5. This organization allows extraction of data on a record in a single server request and output to users as they need it.

4.3 Requirements, Installation, and Configuration of RockProp

Before the RockProp application can be used on a computer, it has to be installed properly. First, the machine has to be running Microsoft’s Windows 2000 (or higher) operating system. Since the web application is based on ASP version 3.0 technology, the Microsoft Internet Information Server (IIS) version 5.0 (or higher) has to be installed on the machine. The IIS has to be properly configured to access the application’s web directory.

In order to use the databases, Microsoft’s SQL Server 2000 has to be installed on the same machine. The three databases have to be rebuilt on the SQL Server after its installation. This process requires restoration of databases from their backup files.

Before the middle tier can be used to access the database, it has to be recompiled. When the middle tier establishes a connection to the database server, it addresses it by a server name. Every database server name is based on the computer’s network name. The name of the server
has to be changed in the \textit{GetServerName} function of the \textit{ServerAccess} module (Appendix G). Currently it is set to the “ALEXT” value which was the name of the development machine.

Finally, Appendices B, C, and D contain information about the database roles. These roles have to be recreated manually after restoration of databases. These roles are used to register users and to determine user privileges.
The successful introduction of HTML (Hypertext Markup Language) is largely responsible for the rapid growth of the Internet’s popularity in the mid 1990s. Simple to use, HTML allows anyone to put information on the web in an effective and creative manner. Platform independence is another feature that made HTML popular. An HTML document is a plain text file that consists of ASCII characters. Today, most operating systems on different computer platforms can read plain text without the need of conversion.

HTML consists of elements, delimited by predefined tags. A tag is a keyword enclosed inside triangular brackets (<tag>) and is a processing instruction for the web browser. HTML tags define document headings, paragraphs, lists, links, images, etc. Tags often contain attributes (<tag attribute=”value”>). An attribute may be described as a parameter for a processing the objective of the tag. Elements are the building blocks of HTML. An element consists of an opening tag (<tag>), informational content (set of characters) and a closing tag (</tag>). In HTML the sole purpose of a tag is to instruct the browser on how information inside an element should be displayed on the screen.

HTML was designed for web publishing. Therefore, presentation of information rather than its content is the main concern of HTML documents. HTML is not suitable for data exchange. 
where information and its description are of great importance. XML (Extensible Markup Language) was designed with the intent of keeping the main advantages of HTML while adding the ability to convey informational content.

This chapter is not intended to be an XML guide to any extent. It does not discuss XML’s origin at all or the markup’s syntax and rules in any detail. There are plenty of books on the subject today (for example, Anderson et al., 2000). The objective of this chapter is to present information on XML data format and corresponding XML schema for the intact rock mechanics data. The next section gives a brief overview of XML.

### 5.1 XML and Its Features

XML was developed to describe data. Like HTML, XML is purely text based. It is also composed of elements that consist of opening tags (with or without attributes), information, and closing tags. Unlike HTML, XML does not have a predefined set of tags; users are allowed to create their own tags and attributes that are relevant to and descriptive of the data being shared.

Figure 5.1 provides an example of a possible XML fragment that describes a core sample.

This example will be used to describe important features of XML.

Being a pure text format, XML can be read by different operating systems across different computer platforms. Its simple

```xml
<core>
  <origin>
    <latitude>50.6785N</latitude>
    <longitude>123.8967W</longitude>
    <depth>230</depth>
  </origin>
  <size>
    <length>90</length>
    <diameter>45</diameter>
  </size>
  <material>
    <rock_type>granite</rock_type>
  </material>
</core>
```

Figure 5.1 Example of XML data.
structure allows XML to be read even without a computer, i.e. it is human-readable.

In XML, users are allowed to create custom vocabularies (sets of tags) for any data. The document structure is developed by users as well. It can be of arbitrary complexity, describing data to any desired level of detail. As a result, different disciplines (engineering, physics, accounting, biology, mathematics, medicine, etc.) can have their own sets of tags, tailored to satisfy their specific needs.

Unlike other text files, an XML file is structured. This can be easily observed from the given example (Figure 5.1). Each pair of tags (opening and closing) defines a data block which can contain sub-blocks, pure data (no sub-blocks), or be a part of a block itself. It is easily seen that an XML document’s structure is tree-like (Figure 5.2). Because of this, the data in an XML file can be easily navigated, searched, and extracted.

The tree-like structure of the XML document allows it to be expanded (new data can be introduced). Extensibility is achieved by adding new elements. Such additions do not change the existing data structure. For example, a new element `<mineralogy>` that may contain information about the core’s mineral composition can be added inside the `<material>` element (Figure 5.3). The vocabulary of XML documents is dynamic, i.e. new data parameters can be added without affecting the existing structure.
XML data is self-explanatory. Values are found inside elements, between tags that describe the nature of data. This is also important. As long as tags remain meaningful (descriptive and unabbreviated), it is easy to understand what sort of data a particular file describes.

XML is well-known for its ability to separate informational content from presentation. Presentation, as mentioned before, is a powerful feature of HTML. XML does not have a built-in presentation mechanism. It can, however, use style sheets. A style sheet is an external document that defines how elements in the main document are presented in the web browser; style sheets are usually linked to main documents using a specific tag. Using style sheets the same data can be presented in numerous ways without changing the data file. This is important; only style sheets have to be changed for data to be presented to different audiences.

Finally, XML allows different vocabularies to be mixed. Sometimes it is desirable to include some information from one vocabulary into another. For example, a rock sample description can be included in a geological description of the area. Suppose a vocabulary for a rock sample description already exists and is frequently in use throughout geological and engineering communities. Suppose also that we want to create an XML vocabulary that would describe the geology of an entire area and provide sample descriptions from its various locations. In this

```xml
<core>
  <origin>
    <latitude>50.6785N</latitude>
    <longitude>123.8967W</longitude>
    <depth>230</depth>
  </origin>
  <size>
    <length>90</length>
    <diameter>45</diameter>
  </size>
  <material>
    <rock_type>granite</rock_type>
    <mineralogy>
      ...
    </mineralogy>
  </material>
</core>

Figure 5.3 Example of XML extensibility.
situation, it would be preferable to use the rock sample vocabulary already in use rather than to write a new one.

Two vocabularies can be mixed in an XML document with the use of namespaces. The namespace is a prefix used in front of an XML tag; it is separated from a tag by the colon sign (<rocksample:tag>). Use of namespaces makes it possible to distinguish tags with identical names but of different vocabularies.

The author would like to note that there are a number of specific XML vocabularies that exist today. In fact, their popularity earned them the privilege of being a language. Some better known ones include MathML (mathematics), WML (wireless applications), SMIL (synchronized multimedia), VoiceML (voice recognition).

5.2 XML Schema and Its Role

It was mentioned that XML data is self-describing. Users can develop their own vocabularies, defining elements that portray their data as freely as they wish. Data, however, is intended for sharing. Users have to know what is expected; they have to know what elements appear in a document, in which order they appear, their format and what type and range of values a particular element can accept. In other words, an XML vocabulary has to be standardized.

Currently, there are two ways to describe the structure of an XML document: Document Type Definition (DTD) and XML Schema methods. The first method is an early attempt to provide a solution to the problem. DTD is a sub-section of an XML document. It usually appears as a set of descriptive statements at the beginning of the XML file. Alternatively, a reference to an
external DTD file can appear instead. DTD statements are written in a language known as Extended Backus Naur Form (EBNF). The DTD section defines every element in the document, relationships between elements, and attributes that can be assigned to them (Anderson et al., 2000, p.69-103).

The XML Schema technology is the latest development in XML document definitions. The schema is metadata – data about data. Like a DTD, it describes the components and rules of an XML vocabulary. The syntax of a schema, however, is completely different from that of a DTD; being an XML file by itself, it does not require a separate language (Anderson et al., 2000, p.248, 249). However, since every schema is an XML document, it must conform to a schema of its own. This is where structures and data types necessary for schema development are defined. Essentially, schemas are written in specific XML vocabulary. At first, it might appear that the XML Schema method is more complicated than the DTD approach and fails to provide any rewards. This is not true. Schemas are easy to use and have numerous advantages over the DTDs.

DTDs are difficult to write and understand, and they lack structure. Many people find EBNF to be obscure and hard to use (Anderson et al., 2000, p.249). Schemas which are programmed in XML, have the same comprehensive tree-like structure as an XML document.

DTDs do not support common data types (e.g. integers, decimals, dates, etc.) that are important when interacting with computer applications or databases. DTDs only support a few XML-specific data types other than text. XML schemas support the majority of data types that are used by computer languages today.
DTDs do not provide support for namespaces and do not allow different vocabularies to be mixed. XML schemas provide full support for namespaces. Use of namespaces is closely related to inheritance.

Inheritance is an important concept in modern object-oriented programming languages and is directly related to data classification and hierarchy. DTDs do not support type inheritance; they do not allow derivations of new types based on existing ones. Unlike DTDs, XML schemas provide an inheritance mechanism; new data types can be easily derived by extension or restriction of existing types. Moreover, schemas allow inheritance of data types from other schemas in addition to data types declared in the schema (Anderson et al., 2000, p.250).

Finally, DTDs only support the closed content model. Nothing other than what is declared in a DTD may appear in the document. XML schemas support both the open and closed content models. A schema author decides if an appearance of undeclared data is allowed (Anderson et al., 2000, p.256).

5.3 Benefits to Geotechnical Industry

A number of statements were made about XML’s elegance. Nothing was said, however, about how the geotechnical field would benefit from using XML. The geotechnical community could gain from the development and use of an XML-based geotechnical data format in several ways.

Such a format can significantly improve electronic data exchange. It is possible to cover every aspect of geotechnical data in a single XML file. At the same time, the file does not have to be
large in size; irrelevant data can be easily omitted as long as the data structure is preserved. Use of a single format for data exchange would eliminate unnecessary data conversion steps.

XML allows users to control the level of detail in transmitted data. Using XML can reduce a large amount of repetitive data being passed back and forth between people. A user does not have to worry about system and computer platform compatibility because XML is text based.

XML data can be easily viewed on paper or on a computer screen. A person can read a printout of an XML file without a computer; the data is self-describing with a tree-like structure that is easy to follow. Of course, being an electronic data format, XML files are best viewed on a computer screen. At the same time, there is no need for specific computer applications. A simple web browser capable of recognising XML can be used to view data.

Presentation of XML data can be easily tailored to a specific audience. With the use of style sheets, XML data can be elegantly presented in a web browser. Data can be easily filtered or transformed into HTML or other XML vocabulary in real time during presentation. Use of different style sheets can show different aspects of data without changing the data file. This allows data to be presented with a specific level of detail, suitable for a specific audience (engineers, geologists, management, etc.).

Use of the XML format can significantly simplify data exchange between different computer applications on different platforms. XML permits data exchange between different application types (mapping, slope stability, finite element, etc.) in a single format. Different departments can utilize a simple file; applications can simply access relevant file sub-structures, extract data from files or write to files.
5.4 Relevance to RockProp

This presentation on XML would not be complete without mentioning benefits of an XML data format to the RockProp project. XML data would greatly simplify future data submission to the database by users. The submission format can be standardised using XML. Using the XML schema, the file can be validated by a user before submission. The data in an XML form can also be easily validated automatically on the server after it has been received, and it can be inserted automatically into a database.

5.5 Design Requirements

There were two main requirements for the design of the XML format and corresponding schema. First, the XML format for the intact rock mechanics parameters should be designed such that it can be extended to include other geotechnical parameters and types of data in the future. Second, the XML schema for the format has to be easy to maintain and extensible.

5.6 Design of the XML Data Format and the Schema

The design of the XML data format and the XML schema for the intact rock properties was performed in three stages: development of the overall geotechnical data structure, design of the data format, and implementation of the schema. The first stage was a key step for the subsequent XML format development stage. It ensures that the XML format can be extended in the future to accommodate other geotechnical parameters.
5.6.1 Geotechnical Data Structure

The geotechnical data structure, used as a guide for the design of the XML format, is shown in Figure 5.4. The tree-path to the intact rock engineering data is traced with black rectangles. White rectangles designate other possible data categories. Their main purpose is to show how the intact rock engineering data relates to the rest of the geotechnical data.

Figure 5.4 shows that the entire geotechnical data is first classified by its spatial and physical origin. The two main materials that geotechnical engineers deal with are rock and soil. The rock mass category is used to classify material that is spatially located in a rock mass. The other category shown (soil) might be used as a starting point in a soil classification scheme.

The rock mass category is further classified based on the category of data obtained from the rock mass. The primary concern of this thesis has been the material properties category. Other subcategories in the same class might include structural data or stress conditions. These are not considered in the current development.

![Geotechnical Data Structure Diagram](image-url)
The *material properties* data can be subdivided further into *intact* and *in-situ* subcategories based on the testing settings or procedures used to obtain actual data. Data within the *intact* subcategory is organised on the basis of testing type. These categories are equivalent to the data sources described in Chapter 2 (also listed in Table A.1, Appendix A).

The geotechnical data structure developed in this section is simple. However, it provides insight on the organization of geotechnical data. The purpose is not to develop the most sophisticated structure but to make sure that the developed XML format is extensible. The subsequent development of the data format for the intact rock parameters is based on the presented structure.

### 5.6.2 XML Document Format Design

The process of the XML format design was straightforward. Major tiers (subgroups) of the document closely follow the structure presented in the previous section (Figure 5.4). The data

```xml
<geotechnicalData>
  <location>…</location>
  <rockMass>
    <dataCategory class="material properties" ReferenceID="1">
      <location>…</location>
      <materialDescription>…</materialDescription>
      <properties class="intact">
        <property type="compressive strength">…</property>
        <property type="tensile strength">…</property>
        …
      </properties>
    </dataCategory>
  </rockMass>
  <references>
    <reference type="book" referenceID="1">…</reference>
    <reference type="journal" referenceID="2">…</reference>
  </references>
</geotechnicalData>
```

*Figure 5.5* Skeleton of the XML document.
parameters at the bottom of the structure, inside the intact subgroup, were arranged according to the data sources described in Chapter 2; data tag names were chosen so that they comprehensively describe themselves. Nevertheless, several interesting issues arose; some needed to be resolved, while others influenced the format design itself.

The crude skeleton of the document is presented in Figure 5.5. As shown, the \texttt{<geotechnicalData>} tag has become the root of the XML document. It is immediately followed by \texttt{<rockMass>} tag (the \texttt{<location>} tag will be discussed below), which in turn is followed by the \texttt{<dataCategory>} tag containing \texttt{class="material properties"} attribute. The rock mass category is designated with a separate tag. The material properties subgroup, however, is designated with an attribute; property class (intact) and each separate property (compressive strength, tensile strength, etc.) are implemented in a similar manner. This illustrates one of the issues with the XML format design; similar structures can be implemented in two different ways: use of tags or use of attributes.

The rock mass subcategory was designated with a separate tag because very few subcategories of such a broad spectrum can be equivalent to it, only soil, for example. Whenever an XML file describes rock material, the \texttt{<rockMass>} tag is always expected. The subsequent subcategories, however, might not exist. If an XML document describes, for example, only structural data then the material properties category would not appear in the file. To make the structure flexible, such categories are implemented as attributes.

Two issues surfaced from the implementation of referencing by the geo-coordinates of a data point: coordinate precision and format. A rock mass can cover a large area. Therefore, a single pair of geographical coordinates provides an arbitrary and vague point of reference. When a
rock sample is described, the latitude and longitude accurately describe the sample position within the rock mass. They, however, may not be always available in which case the set of coordinates for the rock mass would provide at least a sample’s approximate location. Therefore, being useful, both sets should accompany data. To accommodate them, the <location> tag was introduced for the entire record (before the <rockMass> tag) and for every class of data category (inside the <dataCategory> tag). This tag provides location reference on the global (rock mass) and local (sample) levels.

The second issue concerning geo-coordinates is related to their format. There are two popular latitude and longitude formats in use: decimal degrees and degrees-minutes-seconds formats. For example, the longitude of 45°30’00”W is written as -45.5 degrees in the first format and -0453000.00 in the second. The second format is often used in geographical information system (GIS) applications, and at first, it might seem to be a good way of format presentation. However, this standard is very hard to enforce and operate within various computer applications. The RockProp database stores coordinates in the decimal form. It was decided to use the decimal format for geographical coordinates in the XML format and also introduce a special format attribute with the value of “dd.dd” (decimal degrees). The presence of this attribute clearly shows which format is in use. Nevertheless, other formats can be easily implemented in the future. Indicating the format attribute a particular value (for example, “dddmsss.ss” for the second format) provides information on which format is in use.

The placement of measurement units for data parameters was another concern during the design. Measuring units are implemented elegantly by introducing an m_unit attribute within each data
tag that required them. These attributes have appropriate values ("MPa", "km/s", "mm", "MN/m^3", etc.) that correspond to the data tags they describe.

The placement of reference information for the data is an issue as well. Information is referenced on the data category level. First, a separate set of tags were introduced inside the \texttt{<dataCategory>} element. This soon proved to be inappropriate; having several sets of properties (material properties, stress condition, etc.) from one source would force a reference element to be repeated inside each \texttt{<dataCategory>} element. This would make reference data redundant, unnecessarily increasing file size. The problem is resolved by placing all reference information at the end of the document (the \texttt{<references>} element), and referring appropriate references through use of IDs (\textit{ReferenceID}, Figure 5.5). This is similar to having primary and foreign keys in database tables.

Please, refer to Appendix H for the complete structure of the XML document as well as several examples of data files.

5.6.3 Design of the XML Schema

Development of the XML schema is closely related to the development of the XML format. Schemas describe XML files and, therefore, depend on them. The schema design process includes development of schema organization strategy and coding of document structure and constraints.

The schema consists of four separate documents: main schema file, general data types file, reference data file, and intact property types file. The main schema file (\textit{schema.xsd}, Appendix I) is the starting point in the format definition. The main structure of the XML document is
defined in this file. The second file (*general_data_types.xsd*) contains definitions of descriptive elements and their component parts: location and material description sections. The reference data file (*reference_data.xsd*) defines XML structures relevant to the reference information. Definition of intact properties and parameters are located in the last file (*intact_property_types.xsd*).

The subdivision of schema among separate files has advantages over having the entire schema in a single file. First, the separation greatly simplifies the maintenance of the schema. It is easier to work with a smaller file than with a large one. Use of different files allows modifications to be performed on particular structures rather than on the entire schema. Having the schema separated among several files also facilitates extension of the schema. New files relevant to new structures can be easily added in the future without changing the entire schema.

The complete printout of the developed schema is given in Appendix I.

### 5.7 Conclusion

Even though XML is a relatively new technology, it has achieved an important and increasing role in electronic data exchange in many business areas. Any type of data with any level of detail can be described with XML.

At present, there is no indication of XML format development for geotechnical data. This work is an early step in this direction covering, only one small aspect of geotechnical engineering. In order for this work to be continued, the developed XML format and schema have to be reviewed.
by other specialists in the field. *RockProp* will become the first “real” application for the proposed format.
6.1 Summary

Rock engineering often has to deal with problems for which limited information is available. Geotechnical investigations are expensive and labour intensive. It is common to perform a preliminary analysis based on information obtained by previous investigations in the vicinity of a new project. Access to such information is important during a preliminary design stage.

This research attempts to provide practical means for the engineer to deal with data-limited problems in rock mechanics. A web-accessible database for geotechnical data will provide a point of reference for engineers around the world. The author has developed and implemented an extensible relational database for the intact parameters of rock. He has also developed a web-based interface for accessing, searching, and viewing the data in the database. The combination of the database and the web interface comprise RockProp, an Internet application, accessible by anyone who has web access.

The second stage of the research included the design and development of an XML format and an XML schema for intact rock parameters. XML is a promising format for electronic data communication. It allows creation of data vocabularies of desired complexity, which remain easy to use and understand. XML data is self-describing and is easily interpreted by users. XML can be adopted as the data-sharing standard between different computer applications,
running on a variety of operating systems. During this research such an XML format has been developed, as well as a schema for its description and enforcement.

6.2 Future Development

Further developments of RockProp can be carried out in two principal directions: improvements to functionality and accumulation of data.

6.2.1 Functionality Improvements

RockProp’s search functionality can be improved by implementing a search techniques that require minimal user input. At present, the application contains five different search methods, some of which require extensive user input. The project would greatly benefit from the development of a more complex search method based on a simple user input, much like the ones used by many of the popular Internet search engines.

Graphical searches would make finding information easier as well. For example, users would generally prefer to indicate their area of interest by locating it on a map. Development of a search procedure that allows users to point and click on the world map would make the application more intuitive. Showing record locations on the map after a search would also help users to visualize where their data is coming from.

Eventually, it is expected that the RockProp database will contain many thousands of records. Once searches start to return hundreds of records, it will be difficult to navigate through the results. The addition of statistical analysis tools would make it much more effective to deal with voluminous counts of data.
Many users are accustomed to working with specific applications for data analysis. Currently, the only possibility of exporting data from RockProp is to copy and paste values. The ability to export specific data into MS Excel file format or into a simple comma-delimited file would be welcome additions to the interface.

At present, in order to enter data into the database, an administrator has to use the SQL Server’s Enterprise Manager, which is an inconvenient approach. RockProp needs an elegant and functional user interface for data entry.

### 6.2.2 Data Accumulation

At present, the RockProp database contains 77 records obtained from the book by Jackson et al. (1995). This small amount of data is not enough to allow RockProp to be successfully introduced on the Internet. More data has to be entered into the database for it to become useful. This is a crucial feature that must be implemented in the project’s near future.

It was mentioned before that data from mining and geotechnical companies would become a valuable addition to RockProp. In order to be able to accept this data, a set of referencing parameters will have to be chosen, and appropriate additions will have to be made to the database structure of the reference section.
REFERENCES


