**XQuery**  
Welcome to XQuery! The purpose of this application is to design and query a relational database whose data consists of underlying xml storage facilities. An introduction to the problem is given below.

# Overview

There exists a disconnect between our current knowledge employing databases to not only simply store data but to access data in a manner that is relevant to the end users of the data, and between our true knowledge of the underpinnings of such databases and how, when one enters a properly written query in a predetermined language, data is both queried and returned. Being in an advanced database class, this gives us the perfect opportunity to tackle this inherent disconnect between knowledge of how to use a database software such as SQL Server and then the more subtle yet profound knowledge of what happens underneath.

In order to properly study such concepts, instead of directly studying SQL Server, it was our desire to recreate a relational database such that we would have hands on experience with handling all the underlying data storage of a database, which in this case would be XML. Secondly, we would use this relational database to explore the possibility of converting the problem of arbitrary XML querying to the problem of querying a relational database. In this way, we would more aptly and powerfully query xml than was previously possible by harvesting the xml data and properly fitting them into tables with columns that can then be queried relationally. Finally, it was our goal to create a windows program with an accessible yet friendly user interface to act as the presenter and handler of the relational database. Creating a useful design and concept to not only manage data that should be allowed to be boundless in size, as well as creating a language to efficiently gather and present the data to end users, with a program for end users to test, introduces some issues. These issues consist of how to store data in the relational database intuitively in order to then query the data, the syntax of the language that should be utilized, the means to harvest and convert arbitrary xml data into a relational database, and lastly the programming language that should be used to implement the project.

To face these issues we had to decide upon a useful programming environment to create, manage, and test the entire project, and lastly the style of the querying language. We decided it best to emulate the SQL query language style as closely as possible to give us both concrete goals for the different abilities we wanted our querying language to have (these might include the ability to sort, join, select, filter). Moreover, we chose to emulate a SQL querying style that users may need only minimal time in order to get acquainted with our querying language. Therefore, when they are presented with the final project, they can begin testing the various abilities of our query language rather than spend time having to learn the language. It is of fundamental importance for readers to recognize that it was our goal from the beginning to create a relational database foremost, then to convert the problem of querying an arbitrary xml file into the problem of querying a relational database. This would allow us to fully utilize the power of relational databases in tackling the issues of xml querying elegantly and comprehensively. Details on how this was done can be found in the implementation section of this document.

As for the programming environment chosen, we decided to use a programming environment that we were most familiar with, this being the venerable .Net and more specifically C# with its many invaluable libraries, of which we will be making ample use not only to create the GUI with speed, but to ease the many parsing issues that will present itself when trying to query XML. To some, parsing may be thought of as the major component of our project, but to us we feel that allowing .Net to handle the parsing will allow us to focus on the querying mechanics for which this project was duly designed. This will give us more time to tackle and emulate more of the querying concepts and less of the drudgery of xml parsing. What follows is the breadth of our final project report including a detail of the problem to be solved, features of the application, and implementation details.

# Detail of the Problem

Briefly introduced in the overview section of this proposal, the problem we are tackling with this project of creating an XML Query Language is the large disconnect between our understanding of how a database, such as SQL Server, takes arguments in the form of a language style understood by English speakers (such as SELECT, FROM, WHERE), then efficiently queries an underlying storage unit to return data in a format desirable to the user. The presented data could be sorted, grouped, joined across multiple data units or tables, filtered through various constraints, and although the underlying database is stored in static many times useless forms, when it is queried and joined across multiple tables with filters imposed on the query, the data that is returned is deeply meaningful and thoroughly enlightening for users as they ascertain in far greater ways perhaps subtle trends that their products have followed amongst their customers or even as they maintain employee payroll information with the added nicety of being able to track with accuracy and confidence the history of pay raises.

It is obvious that being able to store data in a database and query the data in a form meaningful to the users creates the entire beauty of the concept of a database itself. Though one can study the many esoteric database concepts of data mining, concurrent transactions, query optimizations and the like; understanding the rudimentary concepts of a database, which consists entirely of storing data and presenting data in as flexible a way as possible, is of utmost precedence. This project desires to study just that, and we intend to accomplish the goals of understanding how a database works by creating and managing our own data storage facility as XML files, designing a language easily understandable to end users, and creating the functions of the language necessary to perform adequate querying across the entire breadth of the data, which will most assuredly consist of multiple tables. We plan to emulate the SQL querying language as closely as possible, while managing ourselves the underlying XML storage facility.

Past attempts at creating an XML Query Language can be found in a project developed by Eric Liskow entitled *A New XML Query Language* (cs.hbg.psu.edu/comp519). Though his project was mainly a success, the majority of his efforts, and indeed many sleepless nights, were spent designing a means to effectively parse the xml data in response to the queries issued by the query language. Because we chose to utilize .Net and C# with its many useful libraries that ease creating and managing XML data sources, less work will be done in handling the less meaningful task of parsing, and more time can be devoted toward creating a truly intuitive querying language employing as many language mechanics as possible in attempts to imitate SQL. These include being able to select, sort, filter data, group data, join data with multiple styles of joins, and the like. What follows is a look at the features of the application including a cursory explanation of the user interface.