XQUERY DATA MINING TOOL FOR CAMPUS SECURITY

A Project

Presented to the faculty of the Department of Computer Science
California State University, Sacramento

Submitted in partial satisfaction of
the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science

by
Rohit Surve

SPRING
2015
XQUERY DATA MINING TOOL FOR CAMPUS SECURITY

A Project

by

Rohit Surve

Approved by:

__________________________________, Committee Chair
Meiliu Lu, Ph.D.

__________________________________, Second Reader
Jun Dai, Ph.D.

__________________________________
Date
Student: Rohit Surve

I certify that this student has met the requirements for format contained in the University format manual, and that this project is suitable for shelving in the Library and credit is to be awarded for the project.

______________________, Graduate Coordinator
Jinsong Ouyang, Ph.D.  ____________________________________________  Date

Department of Computer Science
Abstract

of

XQUERY DATA MINING TOOL FOR CAMPUS SECURITY

by

Rohit Surve

In recent years, the process of obtaining information from historical data has gained importance. Data mining and warehousing techniques are used to extract useful data from aggregated data making information retrieval easy and accurate for future predictions. This project involves the design and implementation of a Data mining tool to analyze the information in XML format of the crimes that have occurred on different university campuses in the United States. The main aim of this tool is to extract data stored in separate XML documents and to provide summarized information that will help students in determining the safety level of a campus before accepting an admission. The tool will also act as a source of information for the non-student users like police officers in determining the crime rate across campuses and help them in implementing necessary measures to reduce the crime rate to ensure campus safety. The aggregated dataset is obtained from the U.S department of education website. This data contains both the on campus and off campus crimes that have been reported across various university campuses every year. The tool provides users with statistical data as well as different types of charts for analysis. The tool also provides the results obtained from Apriori algorithm in the form of association rules, which is a pattern that states when an event occurs, another event occurs with certain probability. The main tasks involved in this tool include creation of the database schema, importing of data in XML format into the database, development of SQL stored procedures using XQuery, plotting of
charts, identifying use case scenarios for different crime reports and the implementation of the Apriori algorithm. The front end of the tool is a web application developed in C#.net using a three-tier architecture pattern and the back end database is implemented using Microsoft SQL Server.

_______________________, Committee Chair
Meiliu Lu, Ph.D.

_______________________
Date
I would like to thank Dr. Meiliu Lu for being my project advisor. She has extended tremendous support in the execution of this project and helped me in understanding the approach for application design and development. Her valuable feedback and guidance helped me in the application development and it served as real time experience for product development.

I would also like to thank Professor Jun Dai for his efforts in reviewing the project report and for giving useful feedback.

Finally, I would like to thank my parents and my sister for believing and supporting me throughout the master’s degree program.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Acknowledgements</th>
<th>vii</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Tables</td>
<td>x</td>
</tr>
<tr>
<td>List of Figures</td>
<td>xi</td>
</tr>
<tr>
<td>Chapter</td>
<td></td>
</tr>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Goal of the Project</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Motivation</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Research</td>
<td>2</td>
</tr>
<tr>
<td>2. BACKGROUND</td>
<td>4</td>
</tr>
<tr>
<td>2.1 ETL process</td>
<td>4</td>
</tr>
<tr>
<td>2.1.1 Data extraction</td>
<td>4</td>
</tr>
<tr>
<td>2.1.2 Data transformation</td>
<td>5</td>
</tr>
<tr>
<td>2.1.3 Data loading</td>
<td>5</td>
</tr>
<tr>
<td>2.2 Apriori algorithm</td>
<td>6</td>
</tr>
<tr>
<td>2.3 XQuery</td>
<td>8</td>
</tr>
<tr>
<td>2.4 Standard data source template</td>
<td>8</td>
</tr>
<tr>
<td>2.5 Technology</td>
<td>9</td>
</tr>
<tr>
<td>2.5.1 ASP.net 4.0</td>
<td>9</td>
</tr>
<tr>
<td>2.5.2 SQL server 2012</td>
<td>10</td>
</tr>
<tr>
<td>3. SYSTEM DESIGN</td>
<td>11</td>
</tr>
<tr>
<td>3.1 Application characteristics</td>
<td>11</td>
</tr>
</tbody>
</table>
3.2 Database schema ................................................................. 14
  3.2.1 Data collection .......................................................... 14
  3.2.2 Data loading .............................................................. 16
  3.2.3 Data modeling ......................................................... 16
3.3 Web application architecture ................................................ 17
  3.3.1 Presentation layer ...................................................... 18
  3.3.2 Business layer ......................................................... 19
  3.3.3 Data layer ............................................................... 19
3.4 Low level application architecture ........................................ 20
3.5 Apriori algorithm application design ..................................... 21
4. IMPLEMENTATION .................................................................. 24
  4.1 Data mart implementation .............................................. 24
  4.2 Apriori algorithm implementation .................................... 31
  4.3 Web application implementation ....................................... 36
    4.3.1 Statistics ................................................................. 36
    4.3.2 Chart analysis ......................................................... 37
    4.3.3 Apriori results ....................................................... 41
    4.3.4 Reports ................................................................... 43
5. CONCLUSION ......................................................................... 44
  Lessons learned ...................................................................... 44
  Future enhancements .......................................................... 45
Bibliography ........................................................................... 46
<table>
<thead>
<tr>
<th>Tables</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Crime report types</td>
<td>12</td>
</tr>
<tr>
<td>2. Chart types</td>
<td>13</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figures</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Data warehouse design process</td>
<td>4</td>
</tr>
<tr>
<td>2. Apriori algorithm flowchart</td>
<td>7</td>
</tr>
<tr>
<td>3. Sample SDST XML data pattern</td>
<td>9</td>
</tr>
<tr>
<td>4. Use case diagram for application users</td>
<td>14</td>
</tr>
<tr>
<td>5. Sample raw XML crime data</td>
<td>15</td>
</tr>
<tr>
<td>6. Campus crime star schema</td>
<td>17</td>
</tr>
<tr>
<td>7. Three tier architecture</td>
<td>18</td>
</tr>
<tr>
<td>8. Low level application architecture diagram</td>
<td>20</td>
</tr>
<tr>
<td>9. Apriori algorithm application class diagram</td>
<td>22</td>
</tr>
<tr>
<td>10. Frequent item set</td>
<td>34</td>
</tr>
<tr>
<td>11. Rules</td>
<td>35</td>
</tr>
<tr>
<td>12. Web application home page</td>
<td>36</td>
</tr>
<tr>
<td>13. Web application search criteria</td>
<td>36</td>
</tr>
<tr>
<td>15. Crime count Doughnut chart</td>
<td>38</td>
</tr>
<tr>
<td>16. Total crimes bar chart</td>
<td>39</td>
</tr>
<tr>
<td>17. Crime trend line chart</td>
<td>40</td>
</tr>
<tr>
<td>18. High and medium severity crime bar chart</td>
<td>41</td>
</tr>
<tr>
<td>19. Frequent item set grid</td>
<td>42</td>
</tr>
<tr>
<td>20. Strong association rules grid</td>
<td>43</td>
</tr>
</tbody>
</table>
21. Reports ................................................................................................................. 43
1. INTRODUCTION

This chapter explains the main objective and the motivation for developing this application tool and the research conducted.

1.1 Goal of the Project

The main objective of this project is the development of a web based application tool for students and university police to determine the crime types and crime trend occurring both on campus and off campus in different university campuses in the United States. This application aims in providing statistical campus crime data to the user as well as help him in predicting the relationship between different crime types by way of association rules generated using Apriori algorithm. The data is distributed among several files. XQuery, which is a SQL query and functional programming language [1], is used to query this XML data and return results that can be used for data analysis. Few functionalities and features of the tool are listed below:

- The tool provides year wise crime statistics for universities that is obtained after querying thousands of records from multiple XML files.
- The tool is designed to narrow down the search process based on the intended user type, the state and the university type for faster retrieval of information.
- Different type of charts help the end user to analyze the retrieved information for different aspects like the crime trend, high and low severity crimes occurred every year. This can help the students in determining the safety level of a campus.
- The Association rules generated by the Apriori algorithm helps in predicting which crime events can lead to other crime events. This can also help the campus police to enforce safety measures both on-campus and off-campus to reduce campus crimes.
- The application users can save reports that are generated from a search in different formats like Word and PDF for future reference.
1.2 Motivation

The main motivation was to determine efficient ways of querying data from multiple XML documents. It is based on the “Standard Data Source Template (SDST) concept” as proposed in the IEEE paper “Mining Association rules from complex and irregular XML Documents using XSLT and XQuery” by Xinwei Wang and Chunjing Cao [2]. The campus crime data is stored in different files based on the location of the crimes and the year it occurred. The XML document structure can be very complex having several attributes, which makes it difficult to generate consolidated results. Powerful query languages like XQuery helps in querying these multiple documents and generate a single simple output in the XML format. The XML result can be efficiently used as a standard input structure to the Apriori algorithm [2] implementation making it generic for any type of future data modifications. The main encouraging factor in developing this tool lies in the understanding of developing a system capable of querying data from XML (Extensible markup language) which has become a standard for data exchange over the web. The use of XQuery and approach based on SDST pattern in the implementation of data mining algorithms [3] to extract useful rules and patterns for data analysis is a motivating factor for this project.

1.3 Research

Research is done to retrieve useful information from files containing structured data to produce consolidated reports based on the campus crimes. It is mainly focused on implementation of business logic in stored procedures using XQuery for querying XML documents. Research was involved in the implementation of the Apriori algorithm [2] and the integration of the results returned from the stored procedures. The significance of Standard Data source template had to be understood well. The use of results in this template form was necessary as an input to the Apriori algorithm as the Apriori algorithm requires the input data
to be supplied in a transaction format. Another important research was to make the data retrieval fast and efficient due to large volumes of data spread among multiple files. Therefore the implementation of XQuery following good coding practices and fine tuning of the stored procedures is done. The web application user interface had to be developed as a responsive UI so that it can be rendered on any device such as laptop, tablet or a mobile phone. This involved understanding the jQuery wrapper to implement the responsive UI. The ASP.net charting controls documentation had to be understood in order to implement the configuration settings for displaying data on charts. In addition, the CSC 177 courseware helped me in understanding the Apriori algorithm as well as important concepts in Data warehousing and Data mining.

The report is organized as follows: Chapter 1 contains introduction to the project. Chapter 2 contains some background information on the basic concepts involved in data warehousing and data mining such as the ETL process, Apriori algorithm and the technology used in the application development. Chapter 3 discusses the design details involved in the implementation of the web based data analysis tool. Chapter 4 contains the implementation of the web application architecture, creation of the database schema, the development of stored procedures using XQuery and implementation of the Apriori algorithm to accept input data based on SDST format implemented in stored procedures. Chapter 5 describes about the lessons learned, conclusions reached and the future enhancements to the application.
2. BACKGROUND

This chapter discusses the basic terms and concepts followed in the development of this tool. It explains the Extract, Transform, Load (ETL) process, the Standard data source template (SDST) and the Apriori algorithm. It also discusses about the technology used as well as the role of XQuery in querying XML documents.

2.1 ETL Process

The ETL process is used in the construction of a Data Warehouse. It consist of 3 stages namely Extract, Transform and Load as shown in Figure 1.

![Figure 1. Data warehouse design process](image)

2.1.1 Data Extraction

The data that is stored in the Data Warehouse is initially extracted from the source systems generating the data. The source systems generating this data may store this data in different formats like CSV or Excel files, XML or simple Flat files. Moreover, this data can be stored with other distinct sets of data making the size of the data source typically huge for querying and analysis purposes. This makes it important to extract the data of interest from this large amount of
aggregated data source. For instance, the data source provided by U.S department of education contains 23 different excel and word files for the campus crimes reported every year. CSV and Excel is not widely encouraged as a standard for data exchange over the web because it is not an efficient way for representing semi structured data and requires pre-processing or post-processing. Therefore, Extensible Markup Language (XML) is used for encoding documents that can be read by humans and can be interpreted by machine or software systems [4].

2.1.2 Data transformation

It is defined as a process to transform the source data into a suitable form, which can be used for querying and data analysis. Transformation may include cleaning of the data and perform lookup operations to create lookup tables as well as transformation of data from one data type to the other. For instance, in this application the only manual transformation required is the transformation of the Excel files into XML files. Since the data is represented in the semi-structured format in XML files, no transformation for any attributes is required.

2.1.3 Data Loading

The data after transformation is loaded into data warehouse. The data loading process can vary based on the type of data to be loaded. For instance, in this application the data has to be loaded manually when it is published by the U.S department of Education [5]. This newly published data contains crime incidents for the previous year. Therefore, this may lead to addition of new records in the dimension tables. The fact table also needs to be updated to reflect the new fact values resulting from the updating of the surrounding dimension tables.
2.2 Apriori Algorithm

The Apriori algorithm is a data mining algorithm which is used for frequent item set mining. It begins by selecting the individual items and counting their occurrence in individual transactions. The algorithm uses level wise search to determine frequent item sets. K item sets from one level are used to explore k+1 item sets in the next higher level. A minimum support count is selected which specifies which item sets to consider as frequent items in the given set of transactions. The item sets which satisfy this minimum support count are considered further by combining each of them into larger item sets and determining their occurrence in the transactions. This process is repeated until the last frequent set of frequent subset of item set is frequent. The frequent item sets are then used to determine the association rules.

An association rule is a pattern that states when an event occurs, another event occurs with certain probability. The selection of association rules is based on the minimum confidence level, which is an indicator, which states that association rules below minimum threshold should be discarded. The association rule is given as \( X \rightarrow Y \) where \( X \) and \( Y \) are non-empty subset and \( X \cap Y = \emptyset \).

The support for a rule is given by \( \text{support}(X \cup Y) \) and the confidence is given as \( \frac{\text{support}(X \cup Y)}{\text{support}(X)} \) [3]. Figure 2 explains the working of the Apriori algorithm in detail.
Figure 2. Apriori algorithm flowchart
2.3 XQuery

XQuery can be defined as an expression language which is designed to query data in XML format [1]. It can be used to query data in structured or semi-structured format. It is designed by the World Wide Web Consortium (W3C) working group and is a recommendation for processing XML documents. The XQuery provides essential features to read and manipulate data in XML format as well as address specific parts of the XML document. It is also known as a functional programming language [1]. This project uses XPath expression syntax provided by XQuery to address specific sections of the XML file documents containing the campus crime data. The XQuery syntax is also used in construction of the XML input for the Apriori algorithm based on the SDST template concept [2].

2.4 Standard data source template

The Apriori algorithm requires the input data to be supplied in a certain standard format. The standard XML document can be very complex. Therefore, the use of a “Standard Data Source Template (SDST) concept “ as proposed in the IEEE paper “Mining Association rules from complex and irregular XML Documents using XSLT and XQuery” by Xinwei Wang and Chunjing Cao [2] can be used to create dynamic XML for an XML document of any complex structure. It is a simple XML document having a pre-defined structure. The benefit of using this XML structure is that it can accommodate the changes in the number of input attributes and still be a valid input for processing the input data.
A sample SDST format is shown below in Figure 3.

```xml
<data>
  <record>
    <CrimeYear>2018</CrimeYear>
    <Drug>9</Drug>
  </record>
  <record>
    <CrimeYear>2011</CrimeYear>
    <Drug>5</Drug>
    <Weapon>1</Weapon>
  </record>
</data>
```

Figure 3. Sample SDST XML data pattern

In the above example in Figure 3, the tag `<data>` identifies the set of crime records, which is the root node. The tag `<record>` represents each record in the transaction set. The individual crime types and their values are identified by the attributes within the `<record>` tag. These attributes are supplied to the Apriori algorithm to generate the association rules.

### 2.5 Technology

This section describes about the technology used in the development of this web based campus crime data analysis tool.

#### 2.5.1 ASP.NET 4.0

ASP.Net is a web application development framework which can be used for developing dynamic websites and web applications that can be hosted over the internet. It is part of the .Net framework provided by Microsoft Corporation. The framework is available in the Visual
Studio 2010 and higher versions of the interactive development environment (IDE). The framework helps in writing clean code by maintain the user interface design and the code behind logic in separate files. The server side code can be written in C# or Visual Basic [6]. The ASP.NET 4.0 framework provides an improvement over the previous framework versions. It provides new features like web.config file refactoring, output caching, multi-targeting to target specific versions of .Net framework and jQuery library [7]. Different application architectures such as the three tier architecture [8] can be used to separate the user interface, business logic and data access into separate layers.

2.5.2 SQL Server 2012

SQL server is a database management system provided by Microsoft Corporation. It is used in developing backend databases for large enterprise applications. It provides features like scalability and performance, security, data warehouse and data mining, Reporting services, business intelligence like Integration and Analysis services [9]. The SQL server versions of 2005 and higher provide support to XQuery language making it possible to query structured and semi-structured XML data. The inbuilt performance profiler also helps in fine tuning stored procedures for performance enhancement. In addition, the SQL server provides a feature to store XML data in the database as it defines an XML data type and provides easy options for backup and restore of database.
3. SYSTEM DESIGN

This chapter discusses the design process involved in developing the data mart, Apriori algorithm and the web application.

3.1 Application characteristics

The main aim of the application is to present to the user different types of crime reports for the crimes reported at different locations both on campus and off campus. This section also describes the different type of end users of the application, the different report types as well as the significance of results displayed which are retrieved from data warehousing and data mining. There are two categories of report types in this application based on their crime location namely on campus and off campus. Although, the same crime types have been recorded at both the location types, it can be a useful indicator for both students and the campus police to determine the campus security at different locations.

**On campus crime:** This category represents the crimes that have occurred within the areas described as on campus by the university. It mainly deals with the students who are involved in illegal activities within the premises of the university and can cause a threat to other students. The crime details from this category can be used by new students seeking admission to evaluate the type of students studying in the university. It can also be an indicator of the university student culture and the lifestyle and help them in deciding to take an admission. The residence hall crime details can help students to determine whether the on campus residence halls or dormitory’s are safe and suitable for living. This will also help the university in enforcing discipline among students. The on campus crime details can help the campus police to implement strict measures and rules to ensure safety of the students while studying on campus.
Off campus crime: This category represents the crimes that have occurred outside the premises of the university such as the nearby areas surrounding the university campus. The crime details from this category can help the students who plan to save on living costs and to stay off campus to evaluate the safety of the neighboring areas. It can also provide the local police authorities with the crime reports and help them in enforcing safety measures both for students and for civilians living around the university campus.

Report types: The application extracts useful information based on the different category of incidents occurred in and around the university campuses. These report types contain crime incident types that are most prevalent and severe with respect to the location where they occurred. The crime types have been organized into different files by the U.S department of education to ensure simplicity and ease of access for analysis. Table 1 lists the different report types and the corresponding crime information it contains in the report.

Table 1. Crime report types

<table>
<thead>
<tr>
<th>Report type</th>
<th>Crime types listed</th>
</tr>
</thead>
<tbody>
<tr>
<td>On campus crime</td>
<td>Murder, Forcible sex offense,</td>
</tr>
<tr>
<td>Non campus crime</td>
<td>Robbery, Aggravated assault,</td>
</tr>
<tr>
<td>Residence hall crime</td>
<td>Burglary, Motor vehicle theft</td>
</tr>
<tr>
<td>On campus arrest</td>
<td>Drug, Liquor, Weapon</td>
</tr>
<tr>
<td>Non campus arrest</td>
<td></td>
</tr>
<tr>
<td>Residence hall arrest</td>
<td></td>
</tr>
<tr>
<td>Residence hall discipline</td>
<td></td>
</tr>
</tbody>
</table>
Charts: The charts provide a graphical picture of the campus crimes based on the report type selected for a particular university. Different type of charts like the Bar chart and the Line chart help the end user of the application to understand the overall campus crime trend and individual crime types for different university campuses in the U.S. The results displayed in the charts are based on the results from data warehousing activities on the campus crime data in XML files. This application contains five different chart types as listed in Table 2. These charts make it easy for the end user to understand and analyze the campus crime details.

Table 2. Chart types

<table>
<thead>
<tr>
<th>Chart type</th>
<th>Data displayed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crime count (Doughnut chart)</td>
<td>The composition of individual crime types for the time period selected.</td>
</tr>
<tr>
<td>Total crimes (Bar chart)</td>
<td>The total crimes reported year wise.</td>
</tr>
<tr>
<td>Crime trend (Line chart)</td>
<td>The year wise rise or fall trend for individual crime type.</td>
</tr>
<tr>
<td>High severity crimes (Bar chart)</td>
<td>The year wise high severity crime count.</td>
</tr>
<tr>
<td>Medium severity crimes (Bar chart)</td>
<td>The year wise medium severity crime count.</td>
</tr>
</tbody>
</table>

Reports: The end user of the application can also download reports in Word or PDF format. The user can use these reports for future reference and record keeping. Figure 4 explains the interaction of the user with the application system.
3.2 Database schema

The database is the heart of any enterprise application. The data and business logic resides in the database. In this project, the entire business logic using XQuery has been implemented and stored in the SQL server database in the form pre-compiled database objects known as stored procedures.

3.2.1 Data collection

The campus crime data is published by the U.S department of education and is available online on their official website [5]. This data is published every year. The data is available as set of multiple excel files. The data from multiple files has to be manually imported into the database in
XML format using XQuery script. The XML file is a semi-structured, extensible and platform independent language for data representation making it a popular means of exchanging data over the internet. Figure 5 displays a sample raw XML data.

```
<data>
  <record>
    <UNITID_P>5480047</UNITID_P>
    <INSTNM>Midwestern State University</INSTNM>
    <BRANCH>Main Campus</BRANCH>
    <Address>3410 TAFT BLVD</Address>
    <City>WICHITA FALLS</City>
    <State>TX</State>
    <ZIP>763082099</ZIP>
    <sector_cd>1</sector_cd>
    <Sector_desc>Public, 4-year or above</Sector_desc>
    <men_total>2402</men_total>
    <women_total>3468</women_total>
    <Total>5870</Total>
    <WEAPON1>1</WEAPON1>
    <DRUG1>3</DRUG1>
    <LIQUOR1>3</LIQUOR1>
    <WEAPON1>2</WEAPON1>
    <DRUG1>3</DRUG1>
    <LIQUOR1>1</LIQUOR1>
    <WEAPON1>1</WEAPON1>
    <DRUG1>3</DRUG1>
    <LIQUOR1>1</LIQUOR1>
    <FILTER1>1</FILTER1>
    <FILTER1>1</FILTER1>
  </record>
</data>
```

Figure 5. Sample raw XML crime data
3.2.2 Data loading

The step in the development of the database involves loading the database with the data. The excel documents are first manually imported entirely into the SQL server database and then immediately exported into XML files using XQuery script. After the XML files are generated, these XML documents are imported one by one using an import query which is given below.

```sql
INSERT INTO DocumentMaster (DocName, DocDate, DocContent)
(SELECT ‘residencehalldiscipline111213’,’2014’, CAST(x AS XML)
FROM OPENROWSET ( BULK ‘C:\FinalProject\CrimeData\residencehalldiscipline111213.xml’, SINGLE_BLOB) AS T(x)
)
```

3.2.3 Data modeling

Data warehouse are constructed using dimensional data models for analysis. This consists of facts and dimension tables. Dimension tables contain unique descriptive information. The fact table is constructed from the dimension tables to generate useful results which can be used for analysis. Every record in the fact table is constructed using a combination of foreign keys, which are the respective primary keys in the dimension tables. In this project, the fact table is constructed which contains important facts such as the total crimes, classification of crimes based on their severity type and so on. The fact table values are generated by joining records from the corresponding dimension tables. The fact and dimension table structure in this project form a star schema as shown in Figure 6. The dimension tables contain distinct values of each crime record from the campus crime dataset.
3.3 Web application architecture

The web application is developed using the three-tier architecture pattern as shown in Figure 7 [8]. The advantage of using such a pattern is that it enables segregation of code, thus enabling loose coupling between the different software components within the application.
3.3.1 Presentation Layer

This is the top most layer and is generally referred as the user interface (UI) of the application. The presentation layer consists of view’s which is developed using a UI designer language like Hyper text markup language (HTML). The presentation layer is responsible to hide the complex business logic from the end user and display the data to the user in interactive ways. The presentation layer or UI of this web application is developed using HTML. A master page file is developed which is the parent container for all the child pages. AJAX control toolkit, jQuery library and cascading style sheet’s (CSS) is used for implementing interactive functionalities such as tabs and for styling. ASP.net charting controls library is used for displaying different charts for different types of data.
3.3.2 Business Layer

This layer is the middle layer in the application. The applications business logic and several other operations are implemented in this layer. This layer provides a degree of independence from the presentation layer since the application logic resides within it. In this project the logic to fetch data for binding to appropriate controls, triggering of events and the implementation of Apriori algorithm is done in the business layer. The business layer also contains the different entity objects used within the application.

3.3.3 Data Layer

The data layer is the lowest level layer which contains the logic to access the database. This project consists of a data access layer in the ASP.net application that exposes application programming interface’s (API) to the business layer. These API’s interact with the database by invoking stored procedure calls to the database. The logic to query the XML campus crime files is written in the stored procedures using XQuery. These stored procedures reside in the SQL server database in pre-compiled form as database objects.
3.4 Low level application architecture

Figure 8 explains the low-level architecture of the system. The US department of education contains campus crime data in CSV format. These CSV files are extracted, transformed to generate 23 XML documents. These documents are loaded using SQL scripts into the application.
database. The application receives the search criteria inputs from the user. The application database is queried using XQuery to generate an XML input for the Apriori algorithm. The Apriori algorithm generates association rules based on the XML input. The application database is also queried to perform some data warehousing to generate results for charts. The results from the Apriori algorithm and data warehousing are stored in the form of reports for record keeping.

3.5 **Apriori algorithm application design**

The Apriori algorithm which is a data mining algorithm is implemented in this project using C#.net. The design comprises of a set of classes organized as sub modules as shown in Figure 9. The input items for the algorithm are represented by the Item class. It contains the attributes for item object which are name and support. The support property holds the occurrence count in transactions for the item. The AprioriAlgo class contains the implementation of the Apriori algorithm. It contains methods for generation of candidate sets, identifying frequent item sets, calculating the support, calculation of confidence and identifying association rules. The SorterFascard class is used to sort the transactional elements in the array whereas the validate class is used while generating strong rules from the minimum confidence value supplied. Dictionary and observable collection classes are used to perform different operations like updating and deleting the intermediate results on the collection of items for generating frequent items. The Result class contains the observable collection of the Apriori algorithm results like frequent item sets and association rules. The project module also contains interface classes like the IAprioriAlgo to hide the implementation details and to enforce interface based decoupling making the components loosely coupled.
The different components in the class diagram as shown in Figure 9 are explained below:

**Item**: The item class is used as an observable collection of inputs to the Apriori algorithm. An instance of the item class contains the item name. It is also used to store the support count for the item.

**Result**: The result class is used to store the output of the Apriori algorithm as a collection of strong rules and frequent item sets.

**IAprioriAlgo**: This class exposes an interface that is an entry point to access the Apriori algorithm.

**AprioriAlgo**: This is the main class that contains the implementation of the Apriori algorithm.

**AprioriFascard**: This class is used to prepare the input for the Apriori algorithm from the XML data returned by the XQuery scripts implemented in the stored procedures.
**ISorterFascard and SorterFascard:** This class is used for sorting items while determining the frequent item sets.

**Validate:** This class is used to hold the intermediate results generated from the Apriori algorithm implementation.

Dictionary, ObservableCollection and ContainerFascard are helper classes that are used to store results in the form of an observable collection of elements.
4. IMPLEMENTATION

This section discusses the implementation of the project specified in the previous chapter. The first step requires the creation of the database in SQL server followed by manual import of the XML files. The development of the Apriori algorithm project module and the three-tier web application architecture is done independently for ease of development. The benefit of isolating the Apriori algorithm project is that it can be compiled into a single assembly which can later be integrated into any application. The new application will have to just use the application programming interfaces (API's) which are exposed by the IAprioriAlgo class. The application is hosted locally on Internet information services (IIS) 8.

4.1 Data mart implementation

The chapter 3 described the star schema that is used to make the data mart. The dimensional tables are created before the fact table. The construction of the dimension tables is based on the values in the campus crime dataset. The data definition language (DML) scripts used for the creation of the dimension tables are given below.

Crime type dimension table

The following is the script to create the crime type dimension table.

```sql
USE [CrimeDB]
GO

CREATE TABLE [dbo].[CrimeType](
    [CrimeId] [int] IDENTITY(1,1) NOT NULL,
    [CrimeName] [nvarchar](50) NULL,
    [CrimeSeverity] [nchar](10) NULL
) ON [PRIMARY]
GO
```
The attribute CrimeId serves as the primary key. The keyword identity (1,1) specifies every new entry to be incremented by 1 and the starting position is also 1.

**Program type dimension table**

The following is the script to create the program type dimension table.

```
USE [CrimeDB]
GO
CREATE TABLE [dbo].[ProgramType](
    [ProgramId] [int] IDENTITY(1,1) NOT NULL,
    [ProgramName] [nchar](50) NULL
) ON [PRIMARY]
GO
```

**University dimension table**

The following is the script to create the University dimension table.

```
USE [CrimeDB]
GO
CREATE TABLE [dbo].[University](
    [UniversityId] [int] IDENTITY(1,1) NOT NULL,
    [UniversityName] [nvarchar](255) NULL,
    [ProgramId] [int] FOREIGN KEY REFERENCES ProgramType(ProgramId),
    [LocationId] [int] FOREIGN KEY REFERENCES Address(AddressId)
) ON [PRIMARY]
GO
```

In the above table created by the script, the ProgramId is the foreign key reference from the program type table and LocationId is the foreign key reference from the address table. The University dimension table contains the distinct list of all the universities. The data from this table is used to populate the drop down lists on the web application user interface.
Address dimension table

The following is the script to create the address dimension table.

```
USE [CrimeDB]
GO

CREATE TABLE [dbo].[Address](
    [AddressId] [int] IDENTITY(1,1) NOT NULL,
    [CampusLocation] [nvarchar](255) NULL,
    [State] [nvarchar](255) NULL,
    [Zip] [int],
) ON [PRIMARY]
GO
```

Campus crime fact table

The campus crime fact table is created by performing aggregation operations on the dimension tables. This is done by performing join operation on foreign key references to the dimension tables. Every entry in the fact table is identified by a unique sequence of foreign key references from the dimension tables. The fact table should be updated from time to time as new data is added into the data mart. A stored procedure which is used to extract facts using XQuery is given below.

```
USE [CrimeDB]
GO

-- ==============================================================
CREATE PROCEDURE [dbo].[spGetSDSTStudentData]
(
    @State  varchar(100),
    @ProgramType  varchar(100),
    @UniversityName  varchar(100),
    @TimeFrame  int
)
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;
    Declare @xml XML
    Declare @xmlData XML
```
DECLARE @Temp1 TABLE
(CrimeYear  varchar(100),
Drug  INT ,
Liquor  INT ,
Weapon  INT )

-- Pie Chart results
DECLARE @PieChart TABLE
(CrimeType  varchar(100),
CrimeCount  int )

-- Line chart results
DECLARE @LineChart TABLE
(CrimeYear  varchar(100),
CrimeTotal  int )

-- Apriori Data
DECLARE @Apriori TABLE
(AprioriData  xml )

--- On campus arrest file

INSERT INTO @Temp1 (CrimeYear,Drug,Liquor,Weapon) SELECT '2009',
a.value('DRUG9[1]', 'INT') AS DRUG1,
a.value('LIQUOR9[1]', 'INT') AS LIQUOR1,
a.value('WEAPON9[1]', 'INT') AS WEAPON1
FROM DocumentMaster CROSS APPLY DocContent.nodes('/data/record') as record(a)
WHERE a.value('State[1]', ' varchar(100)') = @State
and a.value('Sector_desc[1]', ' varchar(100)') = @ProgramType
and a.value('INSTNM[1]', ' varchar(100)') = @UniversityName
and DocDate = '2012' and DocName = 'Oncampusarrest091011'
INSERT INTO @Temp1(CrimeYear, Drug, Liquor, Weapon)
SELECT '2010',
    a.value('DRUG10[1]', 'INT') AS DRUG1,
    a.value('LIQUOR10[1]', 'INT') AS LIQUOR1,
    a.value('WEAPON10[1]', 'INT') AS WEAPON1
FROM DocumentMaster CROSS APPLY DocContent.nodes('/data/record') as record(a)
WHERE
    a.value('State[1]', 'varchar(100)') = @State
    and a.value('Sector_desc[1]', 'varchar(100)') = @ProgramType
    and a.value('INSTNM[1]', 'varchar(100)') = @UniversityName
    and DocDate = '2012' and DocName = 'Oncampusarrest091011'

INSERT INTO @Temp1(CrimeYear, Drug, Liquor, Weapon)
SELECT '2011',
    a.value('DRUG11[1]', 'INT') AS DRUG1,
    a.value('LIQUOR11[1]', 'INT') AS LIQUOR1,
    a.value('WEAPON11[1]', 'INT') AS WEAPON1
FROM DocumentMaster CROSS APPLY DocContent.nodes('/data/record') as record(a)
WHERE
    a.value('State[1]', 'varchar(100)') = @State
    and a.value('Sector_desc[1]', 'varchar(100)') = @ProgramType
    and a.value('INSTNM[1]', 'varchar(100)') = @UniversityName
    and DocDate = '2012' and DocName = 'Oncampusarrest091011'

INSERT INTO @Temp1(CrimeYear, Drug, Liquor, Weapon)
SELECT '2012',
    a.value('DRUG12[1]', 'INT') AS DRUG11,
    a.value('LIQUOR12[1]', 'INT') AS LIQUOR11,
    a.value('WEAPON12[1]', 'INT') AS WEAPON11
FROM DocumentMaster CROSS APPLY DocContent.nodes('/data/record') as record(a)
WHERE
    a.value('State[1]', 'varchar(100)') = @State
    and a.value('Sector_desc[1]', 'varchar(100)') = @ProgramType
and a.value('INSTNM[1]', ' varchar(100)') = @UniversityName
and DocDate = '2013' and DocName = 'oncampusarrest101112'

INSERT INTO @Temp1(CrimeYear,Drug,Liquor,Weapon)
SELECT  '2013',
        a.value('DRUG13[1]', 'INT') AS DRUG12,
        a.value('LIQUOR13[1]', 'INT') AS LIQUOR12,
        a.value('WEAPON13[1]', 'INT') AS WEAPON12
FROM DocumentMaster CROSS APPLY DocContent.nodes('/data/record') as record(a)
WHERE a.value('State[1]', ' varchar(100)') = @State
and a.value('Sector_desc[1]', ' varchar(100)') = @ProgramType
and a.value('INSTNM[1]', ' varchar(100)') = @UniversityName
and DocDate = '2014' and DocName = 'oncampusarrest111213'

-- FOR XML PATH ('year'),root('record')

-- Select the time frame for results
IF(@TimeFrame = 3)
BEGIN
DELETE FROM @temp1 WHERE CrimeYear = 2009
DELETE FROM @temp1 WHERE CrimeYear = 2010
END

-- Grid results

SELECT T1.CrimeYear, SUM(T1.Drug) as Drug, SUM(T1.Liquor) as Liquor, SUM(T1.Weapon) as Weapon
FROM @Temp1 T1 GROUP by T1.CrimeYear

-- Pie Chart
INSERT INTO @PieChart(CrimeType,CrimeCount) SELECT
'Drug', SUM(CAST(Drug AS INT)) FROM @Temp1

INSERT INTO @PieChart(CrimeType,CrimeCount) SELECT 'Liquor', SUM(CAST(Liquor AS INT)) FROM @Temp1

INSERT INTO @PieChart(CrimeType,CrimeCount) SELECT 'Weapon', SUM(CAST(Weapon AS INT)) FROM @Temp1

-- Line Chart
INSERT INTO @LineChart (CrimeYear,CrimeTotal)
SELECT T1.CrimeYear, SUM(Drug) + SUM(Liquor) + SUM(Weapon)
FROM @Temp1 T1 GROUP BY T1.CrimeYear

SELECT * FROM @PieChart

SELECT * FROM @LineChart

-- Apriori Data
DECLARE @AprioriData xml
SET @AprioriData = (SELECT T1.CrimeYear, SUM(T1.Drug) as Drug, SUM(T1.Liquor) as Liquor, SUM(T1.Weapon) as Weapon
FROM @Temp1 T1 Group by T1.CrimeYear FOR XML PATH('record'), ROOT('data'), Type )
INSERT INTO @Apriori(AprioriData) VALUES(@AprioriData)
SELECT AprioriData FROM @Apriori
END

Indexing based on search criteria

In order to improve the result retrieval performance of the stored procedures from the XML data, the stored procedures are split to operate on two set of groups based on the state selected in the search criteria. In this approach, for instance, in order to retrieve the on campus crime data for universities, two separate stored procedures “spGetSDSTStudentData2” and “spGetSDSTStudentData3” are created. The records in the XML documents are pre-sorted as per the state in alphabetical order and stored. The advantage of splitting the search based on state reduces the traversing of individual records for the XQuery statement. The logic to select the correct store procedure to execute is implemented in the DataLayer class in the DataAccessLayer project in the web application.

Populating state list

The following is the script for populating the state list.

USE [CrimeDB]
GO

CREATE PROCEDURE [dbo].[spgetState] AS
SELECT Distinct State from [Address] order by State asc
GO

Populating program type list

The following is the script for populating the program list.

USE [CrimeDB]
GO
CREATE PROCEDURE [dbo].[spGetProgramType] AS
SELECT distinct ProgramName from [ProgramType] order by ProgramName asc
GO

Populating university list

The following is the script for populating the university list.

USE [CrimeDB]
GO

CREATE PROCEDURE [dbo].[spGetUniversity]
(  
@State varchar(2),  
@ProgramType varchar(30)
)  
AS

SELECT UniversityName FROM University WHERE State = @State and ProgramName like '%%' + @ProgramType order by UniversityName asc
GO

4.2 Apriori algorithm implementation

The Apriori algorithm is implemented in a modular way to ensure loose coupling between different classes. The IAprioriAlgo interface class is created to provide a single public API as shown below that can be exposed from the assembly to be used by other programs.

Result ProcessTransaction(double minimumSupport, double minimumConfidence, 
IEnumerable<string> items, string[] transactions);

The functionality is separated into different classes as per requirement. Some of the core code functionalities are explained below.

Generating Level 1 frequent item sets:

The code to generate level 1 frequent items initially determines the count of the number of total transactions. The support for individual items is updated based on their occurrence in all the transactions with the help of the GetSupport() function. A list of frequent item sets is created of
object instance type Item and each item is assigned a name and the support value calculated.

Finally the list of items is sorted based on their character corresponding ASCII values.

```csharp
private List<Item> GetFrequentItemsLevel1(double minSupport, IEnumerable<string> items, IEnumerable<string> transactions)
{
    var frequentItemsL1 = new List<Item>();
    double transactionsCount = transactions.Count();

    foreach (var item in items)
    {
        double support = GetSupport(item, transactions);

        if (support >= minSupport)
        {
            frequentItemsL1.Add(new Item { Name = item, Support = support });
        }
    }

    frequentItemsL1.Sort();
    return frequentItemsL1;
}
```

**Generating candidate set:**

This method is used to generate the candidate set after the identification of frequent item sets. Every item in the frequent items list generated by the function GetFrequentItemsLevel1() is combined with every other element in the list to generate all possible combinations between all the items of the list. This generates a new list of items. This new list is again examined if the combinations of items satisfy the minimum support count. If it does, then the newly generated candidate is added to a candidate list which is a Dictionary.

```csharp
private Dictionary<string, double> GenerateCandidateSet(IList<Item> frequentItems, IEnumerable<string> transactions)
{
    Dictionary<string, double> candidates = new Dictionary<string, double>();

    for (int i = 0; i < frequentItems.Count - 1; i++)
    {
```
```csharp
string firstItem = _sorter.Sort(frequentItems[i].Name);

for (int j = i + 1; j < frequentItems.Count; j++)
{
    string secondItem = _sorter.Sort(frequentItems[j].Name);
    string generatedCandidate = GenerateCandidate(firstItem, secondItem);

    if (generatedCandidate != string.Empty)
    {
        double support = GetSupport(generatedCandidate, transactions);
        candidates.Add(generatedCandidate, support);
    }
}
return candidates;
```

**Generating frequent item set:**

The item set list generated in the GenerateCandidateSet() is again iterated to find the frequent item sets and the support count for this new list is updated and the whole process is repeated in a loop until no more frequent item sets can be generated.

```csharp
private List<Item> GetFrequentItems(IDictionary<string, double> candidates, double minSupport, double transactionsCount)
{
    var frequentItems = new List<Item>();

    foreach (var item in candidates)
    {
        if (item.Value >= minSupport)
        {
            frequentItems.Add(new Item { Name = item.Key, Support = item.Value });
        }
    }
    return frequentItems;
}
```

For example, the result generated by the above method for Alabama State University with minimum support 3 is shown in Figure 10.
Generating rules:

The set of rules is generated from the frequent item sets. A set of subsets are generated using the method GenerateSubsets() for every item set containing more than one item. The list of rules combination generated from the subset list is then stored in an observable collection of class type validate which contains the rule attributes as well as holds the confidence.

```csharp
private HashSet<Validate> GenerateRules(Dictionary allFrequentItems)
{
    var rulesList = new HashSet<Validate>();

    foreach (var item in allFrequentItems)
    {
        if (item.Name.Length > 1)
        {
            IEnumerables<string> subsetsList = GenerateSubsets(item.Name);

            foreach (var subset in subsetsList)
            {
                string remaining = GetRemaining(subset, item.Name);
                Validate rule = new Validate(subset, remaining, 0);
                if (!rulesList.Contains(rule))
                {
                    rulesList.Add(rule);
                }
            }
        }
    }
    return rulesList;
}
```
For example, the result generated by the above method for Alabama State University is shown in Figure 11.

![Figure 11. Rules](image)

Generating strong rules:

The rules generated by the GenerateRules() method are evaluated to determine if they satisfy the minimum support count. In the AddStrongRule() method, the confidence calculated in the previous stages for each possible rule is compared with the minimum confidence. If the calculated confidence is more than the user supplied confidence, then the rule is added to the list of strong rules.

```csharp
private void AddStrongRule(Validate rule, string XY, List<Validate> strongRules, double minConfidence, Dictionary allFrequentItems)
{
    double confidence = GetConfidence(rule.X, XY, allFrequentItems);

    if (confidence >= minConfidence)
    {
        Validate newRule = new Validate(rule.X, rule.Y, confidence);
        strongRules.Add(newRule);
    }

    confidence = GetConfidence(rule.Y, XY, allFrequentItems);

    if (confidence >= minConfidence)
    {
```
{ Validate newRule = new Validate(rule.Y, rule.X, confidence);
  strongRules.Add(newRule);
}

4.3 Web application implementation

This section explains the implementation of the web based user interface.

4.3.1 Statistics

The user when requests for the web application is redirected to the home page as shown in Figure 12. The user is then provided with two types of campus crime data namely on-campus and off-campus.

![Web application home page](image1)

Figure 12. Web application home page

The user when clicks on either on-campus or off-campus link is then redirected to the respective page. The user is then provided with a search box as shown in Figure 13 to search data based on the university.

![Web application search criteria](image2)

Figure 13. Web application search criteria
The user first selects appropriate state based on the university in which he is interested to view the campus crimes data. Next, the user selects the appropriate program type and the university drop down list is automatically populated. The user is then required to select the period in years for which he is requesting the reports. The user may also select the minimum support and confidence level for the Apriori algorithm based on which the association rules will be determined.

After the user clicks on the search button, the statistics tab will be displayed with the crime statistic for the selected university. Figure 14 below displays the crime numbers for different crimes that have occurred on-campus for Alabama State University for a five year period.

<table>
<thead>
<tr>
<th>Crime Year</th>
<th>Drug</th>
<th>Liquor</th>
<th>Weapon</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009</td>
<td>6</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2010</td>
<td>24</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>2011</td>
<td>18</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>2012</td>
<td>28</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2013</td>
<td>20</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>

Figure 14. Crime statistics grid

4.3.2 Chart analysis

Based on the search criteria, the application queries the campus crime XML files and retrieves results. The application displays several charts to make the data representation simple [10]. Some of the following charts are described below.

Crime count chart

The crime count is a Doughnut chart which is used to represent the sum of the total of each type of crimes that have occurred for the university with respect to the time period selected. This chart is used to display which crimes have been more predominant in the university campus.
Figure 15. Crime count Doughnut chart

The query statement implemented in the stored procedure that is used to retrieve results for the Doughnut chart in Figure 15 is shown below.

```
SELECT T1.CrimeYear, SUM(T1.Drug) as Drug, SUM(T1.Liquor) as Liquor, SUM(T1.Weapon) as Weapon
FROM @Temp1 T1
GROUP BY T1.CrimeYear
```

In the above SQL query, @Temp1 is an intermediate temporary table variable.

**Total crimes chart**

The total crimes chart is a bar chart which is used to display the total number of crimes that have occurred on campus and recorded year wise for the university selected. This chart helps the user in determining if the count of the number of crimes has increased or decreased year on year.
The query statement implemented in the stored procedure that is used to retrieve results for the bar chart in Figure 16 is shown below.

```
INSERT INTO @BarChart (CrimeYear, CrimeTotal)
SELECT T1.CrimeYear, SUM(Drug) + SUM(Liquor) + SUM(Weapon)
FROM @Temp1 T1 GROUP BY T1.CrimeYear
```

Crime trend chart

The crime trend chart is a line chart which is used to display the crime trend for individual crimes. The chart displays the rise or fall in the crime and is plotted with respect to the year on the x-axis. This chart provides an excellent representation of the crime trend recorded on an annual basis and helps the user in analyzing the severity of a particular crime.
The query statement implemented in the stored procedure that is used to retrieve results for the line chart in Figure 17 is shown below.

```sql
SELECT T1.CrimeYear, SUM(T1.Drug) as Drug, SUM(T1.Liquor) as Liquor, SUM(T1.Weapon) as Weapon FROM @Temp1 T1 GROUP by T1.CrimeYear
```

**High and medium severity crime chart**

The crime severity is a bar chart that displays the crime count year wise for high and medium severity campus crimes. This chart is useful to the users as it divides the crime types based on their severity in two different charts. It displays the crime count for each individual crime for each year.
Figure 18. High and medium severity crime bar chart

The query statement implemented in the stored procedure that is used to retrieve results for the bar chart in Figure 18 is shown below.

```
SELECT T1.CrimeYear, SUM(T1.Drug) as Drug, SUM(T1.Liquor) as Liquor, SUM(T1.Weapon) as Weapon
FROM @Temp1 T1
GROUP BY T1.CrimeYear
```

4.3.3 Apriori Results

The Apriori algorithm is a data mining algorithm which is used for frequent item set mining. The frequent item set are the distinct crime types that are evaluated. The selection of item set is based on minimum support count selected which specifies which item sets to consider as frequent items in the given set of transactions. All item sets below the minimum support count are discarded.

This means all the items appearing in the frequent item set grid have occurred more number of times than the minimum support count after scanning all transactional records. The grid shown in Figure 19 displays the frequent item sets that satisfy the user supplied minimum support count.
The grid in Figure 19 displays the frequent item sets generated based on on-campus crime records for Alabama State University for a five year period from 2009 to 2013.

![Frequent Items Grid](image)

**Figure 19. Frequent item set grid**

An association rule is a pattern that states when an event occurs, another event occurs with certain probability. The selection of association rules is based on the minimum confidence level, which is an indicator, which states that association rules below minimum threshold should be discarded. The association rule is given as $X \rightarrow Y$ where $X$ and $Y$ are non-empty subset and $X \cap Y = \emptyset$. The support for a rule is given by $\text{support}(X \cup Y)$ and the confidence is given as $\frac{\text{support}(X \cup Y)}{\text{support}(X)}$ [3]. In the association rules grid displayed in the application, if one of the strong rule is given as "Drug $\rightarrow$ Liquor, Weapon" and confidence is 0.6; This means that if a crime of type Drug is recoded, then there is a 60% probability that the crime of type both Liquor and Weapon are also recorded.

The grid shown in Figure 20 displays the association rules based on the frequent item sets which satisfy the user supplied minimum confidence. The rules generated are based on the on-campus crime records for Alabama State University for a five year period from 2009 to 2013.
4.3.4 Reports

The application user can download reports in Excel, PDF or word format as shown in Figure 21 for future reference. These reports contain all the statistics data and charts based on the search criteria selected.

![Strong Rules Grid](image1)

**Figure 20. Strong association rules grid**

**Table: Strong Rules**

<table>
<thead>
<tr>
<th>Rule</th>
<th>Confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drug -&gt; Weapon</td>
<td>0.8</td>
</tr>
<tr>
<td>Drug -&gt; Liquor, Weapon</td>
<td>0.6</td>
</tr>
<tr>
<td>Drug -&gt; Liquor</td>
<td>0.6</td>
</tr>
<tr>
<td>Drug, Liquor -&gt; Weapon</td>
<td>1</td>
</tr>
<tr>
<td>Drug, Weapon -&gt; Liquor</td>
<td>0.75</td>
</tr>
<tr>
<td>Liquor -&gt; Drug, Weapon</td>
<td>1</td>
</tr>
<tr>
<td>Liquor -&gt; Drug</td>
<td>1</td>
</tr>
<tr>
<td>Liquor -&gt; Weapon</td>
<td>1</td>
</tr>
<tr>
<td>Liquor, Weapon -&gt; Drug</td>
<td>1</td>
</tr>
<tr>
<td>Weapon -&gt; Liquor</td>
<td>0.75</td>
</tr>
<tr>
<td>Weapon -&gt; Drug, Liquor</td>
<td>0.75</td>
</tr>
<tr>
<td>Weapon -&gt; Drug</td>
<td>1</td>
</tr>
</tbody>
</table>

![Reports](image2)

**Figure 21. Reports**
5. CONCLUSION

This section describes the lessons that I have learned during the implementing of this project. XQuery Data mining tool for camps security is successfully implemented using the XQuery language. The implementation of several charts and the Apriori algorithm to generate association rules presented useful data making it easier to understand and analyze for the end user.

Lessons learned

The most important lesson that I learned is the extraction of useful knowledge from aggregated data that is stored separately in multiple different files. The campus crime data provided on the U.S department of education website contains thousands of records with multiple attributes. The aim of this project was to extract this data in a suitable and efficient way and to provide it to the Apriori algorithm in an acceptable format.

The use of XQuery for extracting data from XML files was a very good learning experience for me in understanding of how to query XML files for data searching and modification. The presentation of data in the form of charts needed some data warehousing activity to be performed. This helped me in understanding the data pre-processing activity before constructing a Data mart. It also helped me in understanding of how to resolve the data into different dimension tables and how to select and retrieve the facts which are a part of the fact table.

The implementation of the Apriori algorithm to accept the XML data was another programming concept which I learned while implementing the project. Modularizing the code into a single assembly which can be used later in another projects was one of the important concepts which I learned while implementing the Apriori algorithm. The implementation of a three tier layered application helped me in understanding on how to develop enterprise applications and to segregate the different layers so as to maintain a level of loose coupling. It made it clear to me that the loose coupling among the data layer, business layer and presentation layer makes it easy
for the application developers for maintaining the applications and for future modifications. Due to this, the present web application layer can be replaced with a desktop application layer making it a desktop application without making any changes to the business layer and data layer.

**Future enhancements**

The data collection phase is an important aspect in this application. The database needs to be updated every year with the crime data published by the U.S department of education on their website. Therefore, a new tab known as administration needs to be introduced with features that would enable importing of the data into the database via the website. A logging mechanism can also be introduced to track the activity of the users requesting information for a particular university. This can help universities predict the number of students who would be interested in joining the university and help them in taking proactive measures to reduce campus crimes by improving security around their campus. The log information can also help in caching frequently requested data and reports thereby increasing the performance of the website to handle multiple requests.

The business layer can be exposed via services so that the reporting API’s can be consumed by different university websites. Important methods like the Apriori algorithm and the statistics data methods can be used by different universities to obtain useful campus crime information.
BIBLIOGRAPHY


https://msdn.microsoft.com/en-us/library/dd456632%28v=vs.100%29.aspx

[Accessed Apr 1, 2015].