ENERGY DATA ANALYZER SYSTEM

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Basudha Pradhan

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Program:
Software Engineering

May 2012

Fargo, North Dakota

North Dakota State University
Graduate School

Title

ENERGY DATA ANALYZER SYSTEM

By

Basudha Pradhan

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Kendall Nygard
Chair

Simone Ludwig

Anne Denton

Edward L. Deckard

Approved:

May 11, 2012
Date

Kenneth Magel
Department Chair

ABSTRACT

Analyzing the electrical energy load and pricing data is important to make informed decisions about energy needs and regulations. Data from over a decade are publicly available online for many Independent System Operators (ISOs). To analyze such a vast data collection demands efficiency, automation, and querying capabilities. The Energy Data Analyzer System is a Graphical User Interface (GUI)-based application program specific to the New York ISO (NYISO). The analyzer downloads the data files from NYISO's website and stores their contents in MySQL database tables for querying. The analyzer can also dynamically create certain custom queries to extract information from the database tables based on a user's selection. The Energy Data Analyzer System, thus, facilitates the study of electricity markets by empowering users with the ability to query data and to analyze data longitudinally.

ACKNOWLEDGMENTS

I would like to thank my adviser, Dr. Kendall Nygard, for his constant guidance and support. This paper would not have been possible without him. I would also like to thank all the supervisory committee members for their time and consideration.

Also, my sincere thanks to the students in Smart Grid group for their valuable suggestions.

DEDICATION

This paper is dedicated to my beloved husband, Manish; my parents; my sisters; and my brother, who have always loved me unconditionally and supported me no matter what.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Page

1. INTRODUCTION

Studying electricity markets requires analyzing relevant historic and recent data to determine energy-usage patterns; forecast energy needs; or find optimal solutions for energy generation, distribution, consumption, and pricing. For a simple simulation or a simple study, one could work with hypothetical data. However, to get closer to real-world scenarios, it is desirable to work with real data. Moreover, it is also desirable to find an efficient solution to manipulate and analyze data over a period of time in order to make informed decisions about the energy supply.

Many Independent System Operators (ISOs) administering electricity in different regions have made their electricity market and operation data available to the public. One of these ISOs is the New York Independent System Operator (NYISO) which provides online access to a longitudinal collection of daily energy-load and pricing data in Comma-Separated Values (CSV) files [1, 2]. NYISO maintains files for various market and operation data on energy pricing; examples include day-ahead market, hour-ahead market, and real-time market pricing. Similarly, it maintains data on load, such as the load forecast for the current day, next day, and real-time actual load. Throughout each day, new files containing the day's data are uploaded, with timestamps, to the NYISO's website. To study these data, the Energy Data Analyzer System was developed as a GUI-based solution that enables users to selectively download CSV files from NYISO's website and to upload contents from the files into MySQL database tables. Having the files stored as database tables provides an efficient way to extract information by querying, which assists in analyzing the data to make informed decisions. The Energy Data Analyzer System automates the process of downloading files; reading those files; creating databases and

tables to store the data; and creating custom queries, such as calculating average energy load, for different time intervals.

The Energy Data Analyzer System achieves the above-mentioned functionalities by using a MySQL database, an open-source relational database (version 5.5) to create databases, create tables, and run queries. The system interacts with a MySQL database using a plug-in called MySQL Connector Net version 6.4.4 (as discussed in Chapter 2).

Given that the system requirements are all met, upon running the Energy Data Analyzer System, users can interact with the system using the provided GUI. Below are some of the high-level user-interface design requirements for the Energy Data Analyzer System:

1. Users shall be able to select or deselect the load or pricing data files to download.
2. Users shall be able to upload data from the downloaded files into MySQL tables.
3. Users shall be able to select (a) database table(s) from which to query.
4. Users shall be informed of any errors in their selection. For example, notify users if there are no files available for download based on their selection.
5. Users shall be notified when a download operation or an upload operation has been completed.
6. Users shall be provided with helpful hints to navigate through the system.

In addition to these user-interface design requirements, the Energy Data Analyzer System has several desired functionalities. The core system functionalities are downloading files from the NYISO website, uploading data in the files into MySQL tables, and querying the tables to get desired information. Below is a list of high-level functional requirements for the system:

1. Users shall be able to select (a) file(s) for a certain date or a range of dates to download.
2. The selected files shall be downloaded to the user's local machine.

3. The Energy Data Analyzer System shall create MySQL database tables and store data from the downloaded files in the tables.
4. Based on the user's selection, the Energy Data Analyzer System shall dynamically create certain custom queries for the zonal load commitment forecast, the real-time actual load under load data, and the day-ahead market zonal and real-time market zonal pricing data.
5. The result of the query shall be in CSV format which can be opened in Excel for further manipulation.
6. Users shall be notified in the case of an error.

Chapter 2 contains a Literature Review of various technologies and methods used in developing the Energy Data Analyzer System. In Chapter 3, the Energy Data Analyzer System interface design, use cases, and system functionalities are discussed. Chapter 4 demonstrates the utilization of the system based on a use case. Chapter 5 explains limitations and potential future work involving the system. The References are at the end of this paper.

2. LITERATURE REVIEW

The Energy Data Analyzer System provides a Graphical User Interface (GUI) to interact with the NYISO website in order to download selected load and pricing data files. The Energy Data Analyzer System uploads the data from the files into MySQL database tables. The technology background relevant to the above process and the development of the Energy Data Analyzer System are discussed in this chapter.

2.1. Microsoft Visual Studio and C#.NET

Microsoft Visual Studio is an Integrated Development Environment (IDE) that provides a platform for .NET application development. It aids developers in producing quality code by providing built-in tools and controls, such as buttons, text boxes, labels, etc., and features, such as intelliSense. It also provides debugging capabilities.

C# is a programming language based on object-oriented design and was mainly created to be used with .NET development [3]. Development in .NET supports using different programming languages such as J#, Visual Basic, C++, and C# [4]. .NET development offers platform independence, which is made possible by compiling programs written in different languages to a common Microsoft Intermediate Language (IL) [5]. IL does not contain any machine codes native to the computer. Therefore, another compilation is needed to convert IL to machine code using a Just-In-Time compiler (JITer) [6].

Any .NET development requires using the .NET framework which provides a large collection of built-in reusable classes known as .NET framework classes [7]. The Common Language Runtime (CLR) is the core of .NET technology. The CLR is a runtime execution environment responsible for thread execution, memory allocation, and Just-In-Time compilation

[8]. Also, ASP.NET and XML support allows the easier import and export of data, and ADO.NET support allows the capability to work with databases.

The Energy Data Analyzer System was created using Microsoft Visual Studio 2010 premium version [9]. The database it works with is a MySQL database, which is discussed next.

2.2. MySQL and Structured Query Language

MySQL is a popular open-source relational database management system. It is used with sites such as Google, Facebook, and Twitter because of its powerful performance [10]. The SQL in MySQL stands for Structured Query Language (SQL). SQL is a query language to manipulate and query data in a relational database. SQL queries can contain statements, clauses, expressions, and predicates necessary to produce the desired result [11]. MySQL can be downloaded from the MySQL website and can be installed on a computer as a server. The Energy Data Analyzer System uses MySQL Server 5.5 to create and store databases and tables.

2.3. MySQL Connector Net

MySQL Connector Net is an open-source plug-in that integrates Visual Studio and MySQL by providing an ADO.NET driver for MySQL [12]. Developers can create .NET applications that use MySQL databases. Once the plug-in is downloaded and installed, a reference to MySql.Data.dll, which is a class library, can be added to the program. The Energy Data Analyzer System uses MySQL Connector Net 6.4.4 to link with the MySQL database.

2.4. New York Independent System Operator

The New York Independent System Operator (NYISO) administers New York state's energy need and also operates the electricity grid and wholesale electricity markets. It covers

nearly 11,000 miles of high-voltage transmission lines from 500 generators. The load, transmission, and electricity cost are monitored and recorded 24 hours a day, 7 days a week. Utility companies obtain electricity for distribution to their consumers through the NYISO's wholesale electricity market. The minimum purchase necessary is one megawatt, and it can supply power to nearly 1,000 homes [13]. There are 11 energy pricing zones in NYISO as shown in Figure 1 [13].

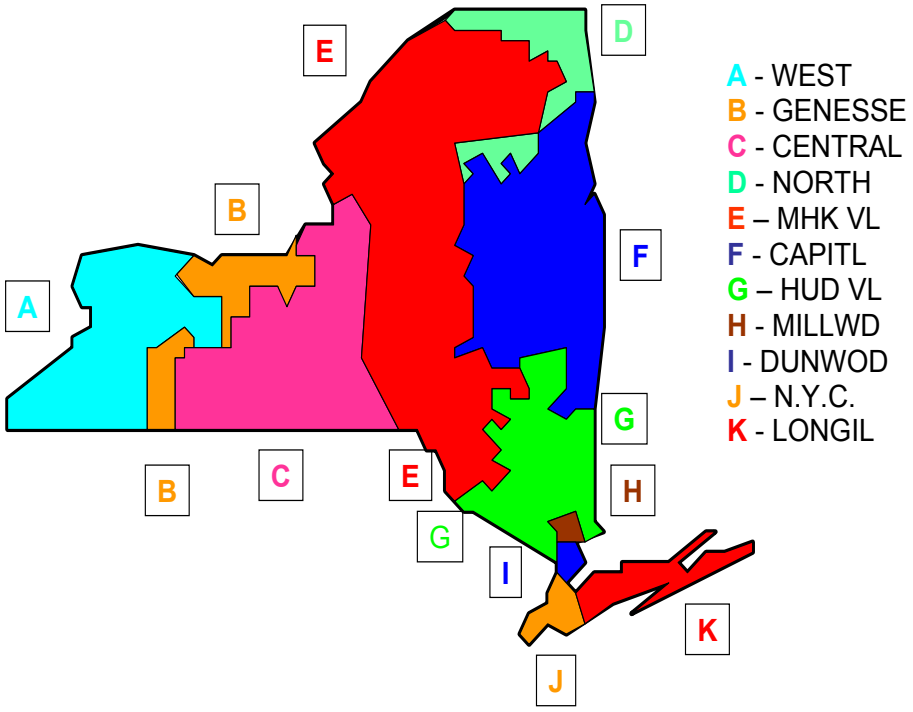

Figure 1. Energy pricing zones.

2.4.1. Locational Based Marginal Price

One of the components of wholesale electricity prices is the energy cost known as Locational Based Marginal Price (LBMP). LBMP can be calculated by the cost to produce energy plus the cost to transport that energy [14].

2.4.2. Day-Ahead and Real-Time Electricity Markets

NYISO operates a day-ahead market and a real-time market [15]. Day-ahead market data provide the next day's operating schedule for generators. By providing an energy forecast in combination with the cost of electricity forecast, the generators are financially motivated to operate as scheduled. However, there can be certain factors that influence energy generation, such as system availability and condition, which can alter the projected load, so NYISO runs a real-time market to account for the changes and to maintain a financial balance.

In the next chapter, the Energy Data Analyzer System is discussed in detail. The system functionalities and methods used to implement those functionalities are included.

3. ENERGY DATA ANALYZER SYSTEM

NYISO offers online access to electricity market and operation data to the general public and NYISO's market participants. The electricity market can include a day-ahead market, a real-time market, and an hour-ahead market. These markets capture energy pricing and energy schedules for the generators based on the market type. These data are monitored and recorded 24 hours a day according to timestamps, grouped by energy pricing zones or some other criteria, and are available as CSV files.

The Energy Data Analyzer System allows users to interact with the NYISO website through its GUI. Users can select the CSV files on energy markets and download them to their own local machines. The system then stores the data in the files as tables in PricingData or LoadData MySQL databases which can be queried to get the desired information. Upon running the Energy Data Analyzer System, the PricingData and the LoadData databases, and the table templates, along with a folder called NYISODB are created, if these did not already exist. Using the Energy Data Analyzer System's GUI, the user can then select various types of pricing and load data files available at the NYISO website. The Energy Data Analyzer System downloads the selected files from the NYISO website to the NYISODB folder. After the download, separate MySQL database tables are created from the table templates to store each file's content. Since, the data is stored as database tables; users can query the databases to get the desired information. The Energy Data Analyzer System also provides built-in custom queries to calculate the average energy load and pricing during various time intervals for the energy zones. Users can calculate the average in multiples of five-minute interval; the maximum is sixty-minute interval, which represents the next hour. The results of the built-in custom queries are provided in the CSV file format.

Figure 2 shows an overview of the Energy Data Analyzer System, where a user communicates with the NYISO website, PricingData, and LoadData databases using the GUI to query and get the CSV file output. The CSV file output can be opened in Excel for further data manipulation, such as adding filters and creating graphs. During the file selection, the user interacts with the GUI to select the CSV files on energy load or pricing to be downloaded. The Energy Data Analyzer System analyzer retrieves the files from the NYISO website and downloads them to the NYISODB folder in the user's local machine. After the files are downloaded, each of the file's content are uploaded into the tables in PricingData and LoadData databases. Users can query the databases and get the results in the CSV file format and manipulate the data using Excel.

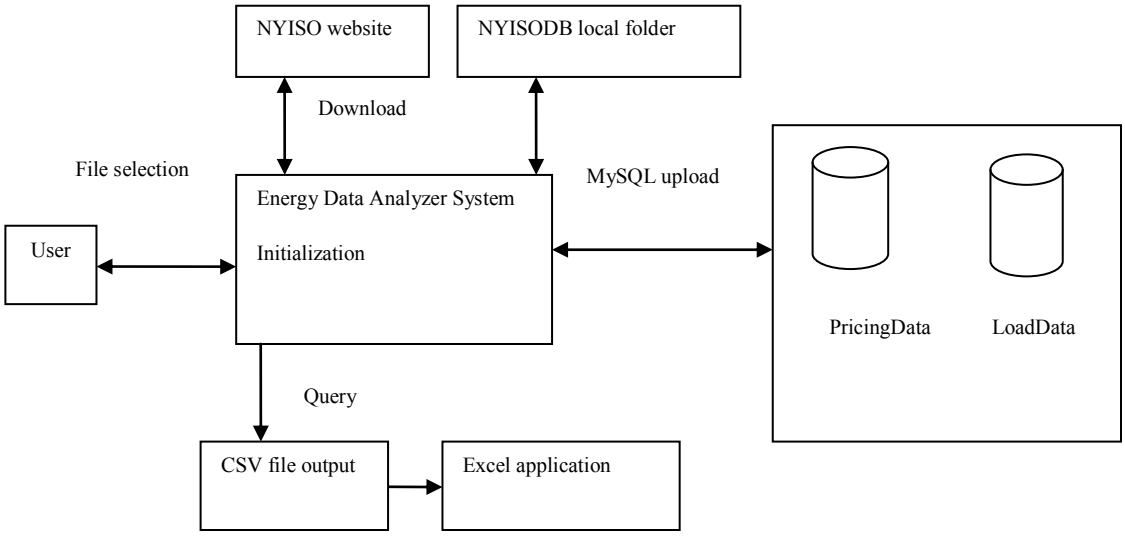

Figure 2. Energy Data Analyzer System overview.

The Energy Data Analyzer System can be divided into five different parts based on its functionalities. These five core functionalities are as follows:

1. Initialization
2. File selection
3. Download
4. MySQL upload
5. Query

3.1. Initialization

Upon running the Energy Data Analyzer System, if databases named PricingData and LoadData do not exist, the system creates the two databases. A folder called NYISODB is created to hold the downloaded energy files in the user's personal folder. Also, there are certain types of files under the Load data tab and the Pricing data tab. To store data from these files, new MySQL database tables are created based on built-in template tables provided in the Energy Data Analyzer System. For example, the NY Load Forecast file may have certain columns, so depending upon the file name; the Energy Data Analyzer System creates a new table similar to one of the template tables having matching columns. There are a total of eight template tables, tablestyle1 through tablestyle8, that are created if they did not exist before the program was run. An alternative was not to provide any template tables, but to create tables dynamically while reading the files. However, some files did not have header rows to create columns in a table, so this approach was abandoned. Each time the program starts; the PricingData and LoadData databases and the template tables are checked. Even if the user accidently deletes the templates or databases, the Energy Data Analyzer System would function smoothly when it runs the next time. Figure 3 shows the Energy Data Analyzer System with the Load Data tab selected. Users can click the checkboxes and press the confirm button to select files to download.

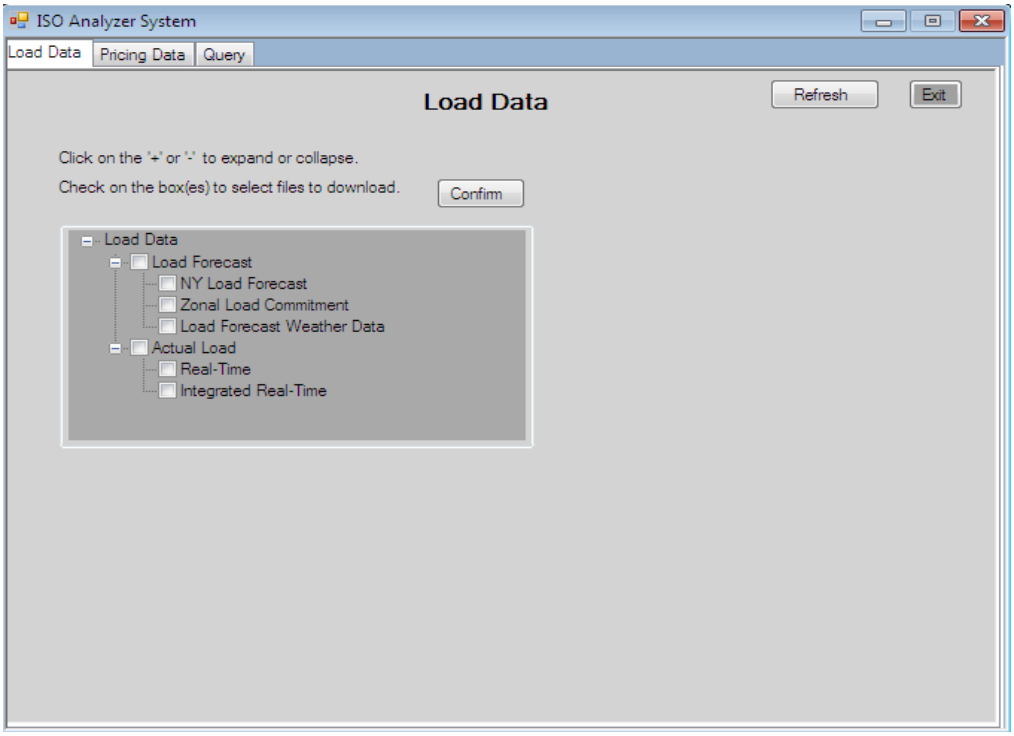

Figure 3. Energy Data Analyzer System screenshot 1.

Figure 4 describes the logic during initialization. The LoadData and PricingData databases, the template tables and the folder to store the downloaded files may or may not exist when the program is run. The initialization logic checks and creates these if they do not already exist.

Initial Condition: LoadData and PricingData databases, template tables, a local folder to hold files to be downloaded may or may not be created.

Logic:
Create a folder named "NYISODB" on a local machine to store downloaded files.
Create the Loaddata and Pricingdata databases if they do not exist.
Create template tables if they do not exist.

Figure 4. Initialization.

3.2. File Selection

NYISO maintains the energy data files by date. For consistency, the Energy Data Analyzer System allows users to select files for a particular date or a range of dates. The Date area is visible only after the user selects at least one file by clicking on the checkbox next to a file type represented as a node in the tree view control and pressing the confirm button to confirm the files to download. Figure 5 shows the files selected for certain dates.

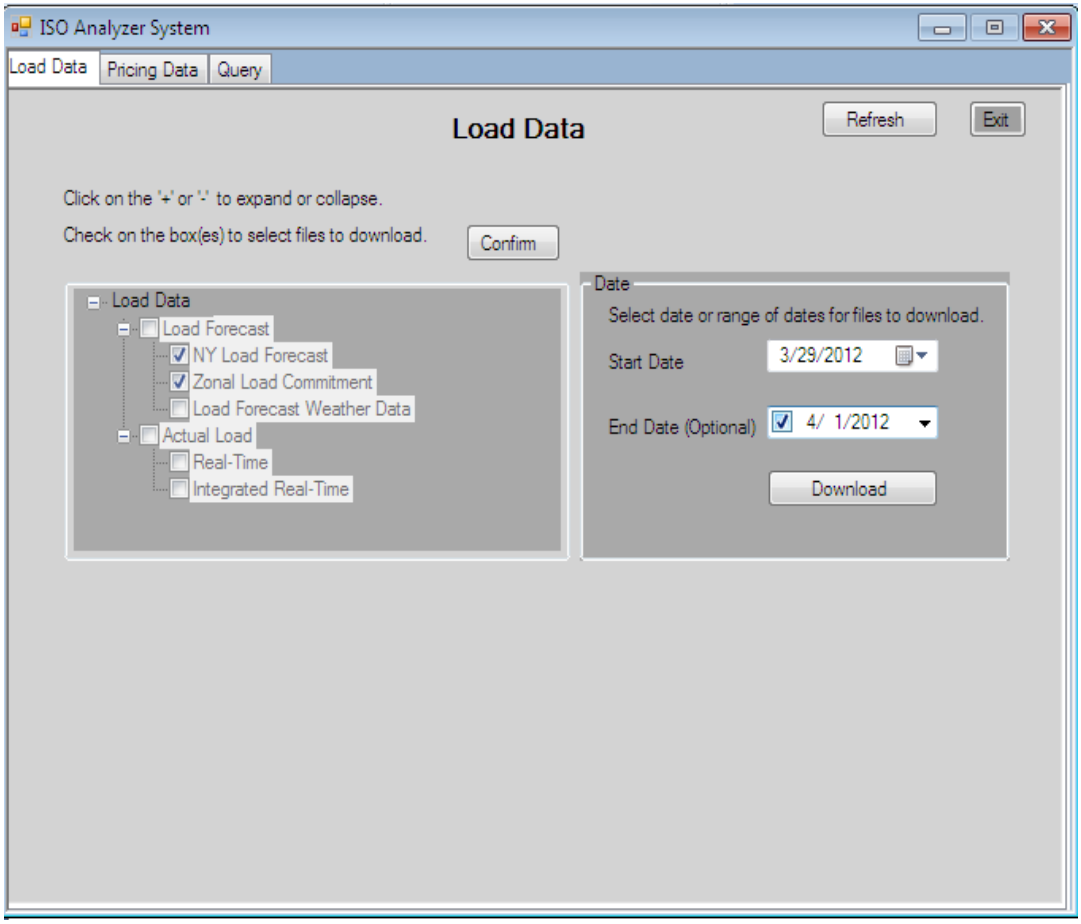

Figure 5. Energy Data Analyzer System screenshot 2.

Each file denoted as a node in the tree view control has a tag property which is used to identify the type of the file. The tag property is later used to create the name of the file to be downloaded. For example, Zonal Load Commitment has a tag property value "zonalBidLoad." This tag property decides which template table to use to create a table to hold data from the file. Also, using a tree view control to display various file types makes it easy for users to select the files. One can simply click on the parent node, and all its child nodes would automatically be selected. When the user confirms the file selection, the tree view control becomes read only, and the date selection area becomes visible. Clicking the Refresh button enables file selection. The Exit button can be used to close the application. Figure 6 is an algorithm of the process during file selection.

Input: a tree view control with its tag property set for nodes in the tree view
Output: list of tags of selected nodes
Initial Condition: a checkbox beside each node representing a file name

Logic:
if parent node is checked, **then**
    check all its child nodes
end
if a child node is unchecked, **then**
    uncheck parent node if checked
end
if the Confirm button is clicked, **then**
    verify at least one of the nodes is checked
end
if checked is true, **then**
    make date selection area visible
    add the tag property value of the selected node to a list
else display a message asking the user to select a file to download

Figure 6. File selection.

A user has to select the files to download for querying before the actual download process can be executed. Users can select the files to be downloaded by clicking the checkboxes next to the file names in the tree view control provided in the Energy Data Analyzer System's GUI. Figure 7 is a use case for selecting files to download.

Name: Select Files to Download

Description:
The user selects files to download by clicking the checkbox next to the file name.

Pre-Conditions:

- The Energy Data Analyzer System is running, and the tree view control to select files is enabled.
- All the checkboxes are unchecked.

Post-Conditions:

- Tree view control is made read-only.

Basic Scenario:

1. The use case begins when the user wants to select files for download.
2. The user clicks the checkbox next to the file type he wants to download.
3. The user clicks the Confirm button to confirm file selection.
4. The use case ends,

Alternate Scenario A:

A1. The user does not click any checkboxes and hits Confirm.
A2. The system informs the user to select files first.
A3. The use case begins from basic scenario step 1.

Alternate Scenario B:

B1. The user confirmed the wrong files and would like to select the right files.
B2. The user hits the Refresh button.
B3. The use case begins from basic scenario step 1.

Figure 7. File selection use case.

3.3. Download

After a user selects the files to be downloaded for a certain date or a range of dates, pressing the Download button validates the dates. The system checks if the selected files are available for download from the NYISO website for those dates. If a file is not available, the user is notified.

In Figure 8, the user has chosen to download Real-Time Actual Load for May 30, 2012. Considering that the program was run sometime on April 14, 2012, there are no such files available for download on the NYISO's website. When a user tries to download the file for May 30, 2012, the system displays the error message shown in Figure 8.

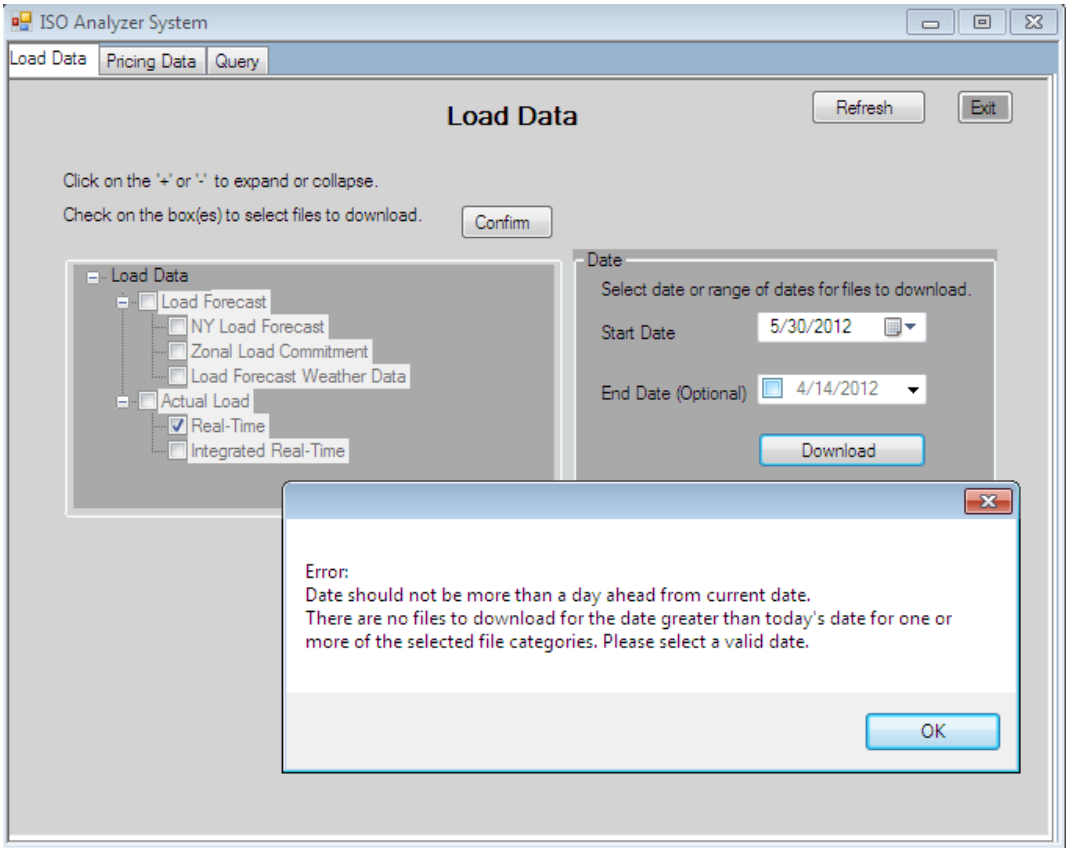

Figure 8. Energy Data Analyzer System screenshot 3.

When a user selects the files to be downloaded for certain dates, the Energy Data Analyzer System creates a list of potential table names by concatenating the date and the tag property value of the selected node from the tree view. These names are later used to create MySQL database table names. For each individual file, a table with data from the file is created. For example, if a user selected the NY Load Forecast file for November 22, 2011, the tag property value of the node representing NY Load Forecast is "isolf." Therefore, the table name for it would be 20121122isolf. The resource file for this particular file on NYISO's website would have the name 20121122isolf.csv.

To download files from the web, the Uniform Resource Indicator (URI) should include the exact file name to be downloaded. NYISO keeps recent (8- or 9-day old) files in the CSV format and older files in zip format. For example, if today is May 4, 2012, then the Zonal Load Commitment file for May 2, 2012, is named 20120502zonalBidLoad.csv. A similar file for March 3, 2012, would be in a zip file named 20120301zonalBidLoad.zip. This zip file would contain all the files for that particular month.

The files are downloaded using the WebClient class found in System.Net namespace [16]. The Energy Data Analyzer System calls its DownloadFileAsync method, which downloads the files without halting the calling thread. Aysnchronous download enables users to interact with the GUI while files are being downloaded in the backend. While creating any GUI application, if there is a thread that would require more time and resources to execute, usually, the task is accomplished by using the Background Worker class which makes the GUI responsive at all times through multi-threading [17].

The Energy Data Analyzer System needs to download files to perform query operations. Instead of Background Worker, an alternative was to use the WebClient class

DownloadFileAsync method which pretty much accomplishes the same thing. The logic behind the file download process is presented in Figure 9.

Initial Condition: Files and dates are selected, and the download button is clicked.

Logic:
if no errors in dates selected, **then**
    Check if the resource file will be in .zip or .CSV
if .zip, **then**
        Create a list of .zip file names to be downloaded by concatenating the date and the node property value of the selected node representing a file
    else
        Create a list of .zip file names to be downloaded by concatenating the date and the node property value of the selected node representing a file
    end
    Create a queue holding the URI for all the file names in the list above.
    Download the files with the WebClient asynchronous download method.

else
    Display a message asking the user to select valid dates.
end

Figure 9. Download.

In Figure 10, the user has selected the Zonal Load Commitment and the Load Forecast Weather Data files to be downloaded for dates ranging from February 29, 2012, to March 15, 2012. After the Download button is clicked and provided there are no errors, the actual downloading of the files starts as indicated by a progress bar near the lower-right corner. Once the download is successfully completed, the user is notified with a label that says “Download Complete.” The progress bar is fully filled in as shown in Figure 10.

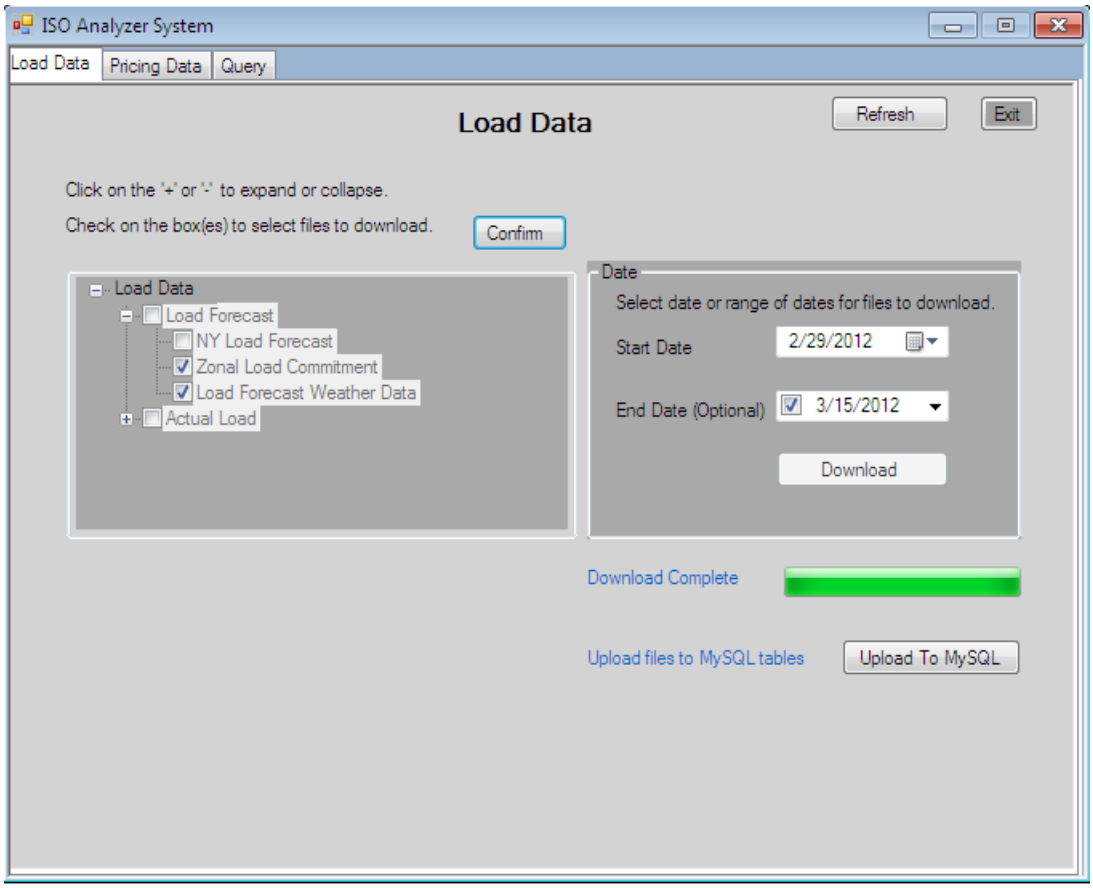

Figure 10. Energy Data Analyzer System screenshot 4.

While downloading, the following scenarios may occur. There are no errors in date selection; files are available for the chosen dates; and then, the download process begins. If there are any errors, then the download process does not begin until all the errors have been corrected. In Figure 11, different use case scenarios for downloading are described.

Name: Download Selected Files

Description:
The user clicks the Download button to download the selected files from NYISO's website.

Pre-Conditions:

- The user selects files to download, and the tree view control to select files is disabled.
- Date selection is visible, and the user has input the dates for which the files are to be downloaded.

Post-Conditions:

- The download button is disabled. The progress bar is visible.
- The download status is shown next to the progress bar; the Upload To MySQL button is visible.

Basic Scenario:

1. The use case begins when a user clicks the download button.
2. The progress bar is updated to show the download status.
3. Also, the label next to the progress bar shows if the download is in process or has been completed.
4. Selected files for the dates are successfully downloaded into a local folder.
5. The use case ends.

Alternate Scenario A:

A1. The user selects invalid dates, such as the end date is earlier than the start date.
A2. The system informs the user to select valid dates.
A3. The user selects dates, and the use case begins from Basic Scenario step 1.

Alternate Scenario B:

B1. The selected dates do not have files available for downloading.
B2. The system informs the user to select valid dates.
B3. The user selects dates, and the use case begins from Basic Scenario step 1.

Alternate Scenario C:

C1. The file download is taking a long time; the user decides to cancel the download.
C2. The user clicks the cancel button.

Figure 11. Download use case.

3.4. MySQL Upload

After the files are successfully downloaded to the local folder named "NYISODB," the Energy Data Analyzer System lets the user know the download is complete using a label that says "Download Complete," or users can also check the progress bar. The next step is to upload data from the files to MySQL database tables by clicking the Upload To MySQL button. As mentioned before, a table based on the file name will be created for each individual file. For the downloaded zipped folders, each zipped folder containing the selected files has to be read. The Energy Data Analyzer System uses DotNetZip [18], a free class library for manipulating zipped files. A reference to Ionic.Zip.dll has to be added in the program to access its classes. Instead of extracting all the files in the zipped folder, only the files that a user selected are extracted to be read. The data from these individual files are transferred into the MySQL database tables.

For example, if a user downloads the Zonal Load Commitment data file for April 4, 2012, there will be a file named 20120404zonalBidLoad_csv.zip, which is the file name of the resource file on NYISO's website. The Energy Data Analyzer System uses DotNetZip to extract the particular file named 20120404zonalBidLoad.csv. The tag property value of the node representing Zonal Load Commitment is "zonalBidLoad," so the table name created for the .csv file is 20120404zonalBidLoad. This new table is created based on one of the table templates matching the columns in 20120404zonalBidLoad.csv.

In Figure 12, the Zonal Load Commitment file for April 15, 2012, has been already downloaded to the user's local machine, and the user has clicked the "Upload To MySQL" button. Upon clicking the button, the Energy Data Analyzer System creates the necessary database tables for each of the recently downloaded files. A new database table containing records from the Zonal Load Commitment file for April 15 is created in the LoadData database.

After the table is created, the user is notified when data upload to the MySQL table has completed using a label that reads “All upload Complete” as shown in Figure 12.

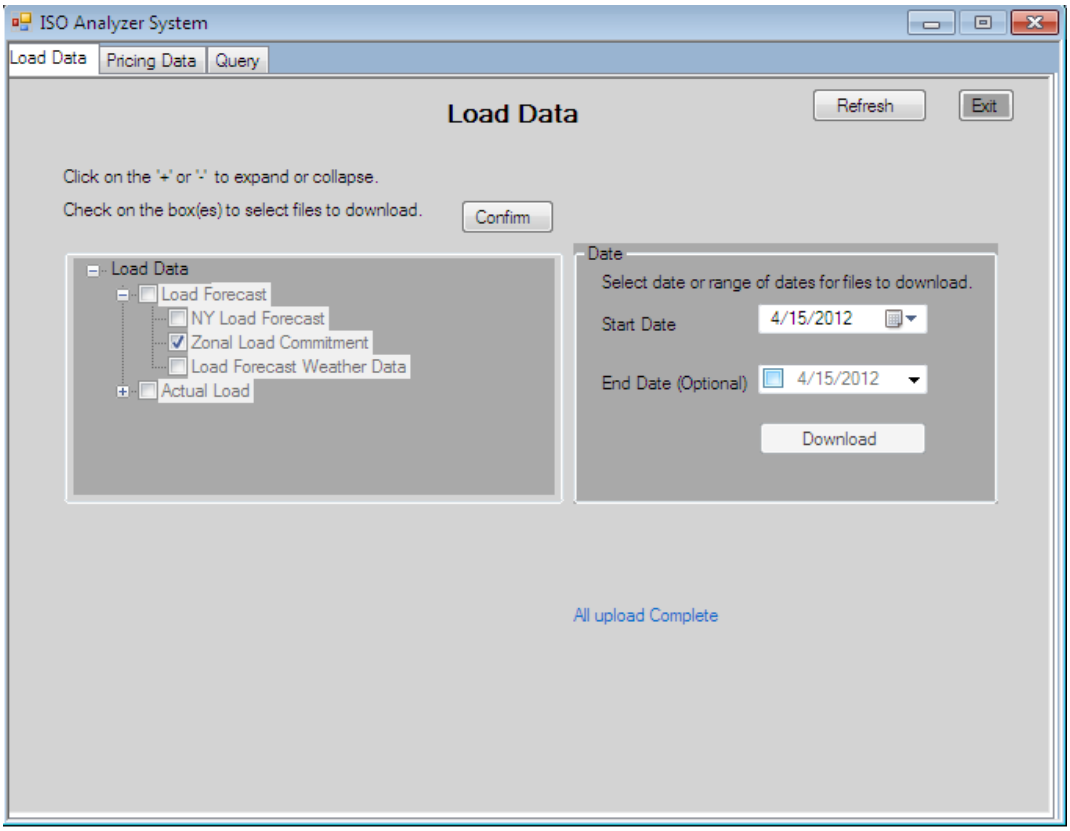

Figure 12. Energy Data Analyzer System screenshot 5.

Data can be uploaded into the MySQL tables following an immediate download, provided that downloading the files and uploading data into tables are executed using the same running instance of the Energy Data Analyzer System. Figure 13 describes the logic for creating and uploading tables into the MySQL databases.

Initial Condition: Files for certain dates selected for download have been downloaded in the NYISODB local folder.

Logic:
For each selected file
Identify the database to which it should belong.
Identify the table template.
Create a new table in the identified database from the table template.
Copy data from a .csv file to the newly created table.
end

Figure 13. MySQL Upload.

Figure 14 is a screenshot of tables uploaded to the MySQL database. For example, to view the tables uploaded into the LoadData database, Visual Studio 2010 IDE's Server Explorer has been used to establish a data connection to the LoadData MySQL database. The numeric portion of the table names represents the date, and the rest represents the type of data it holds.

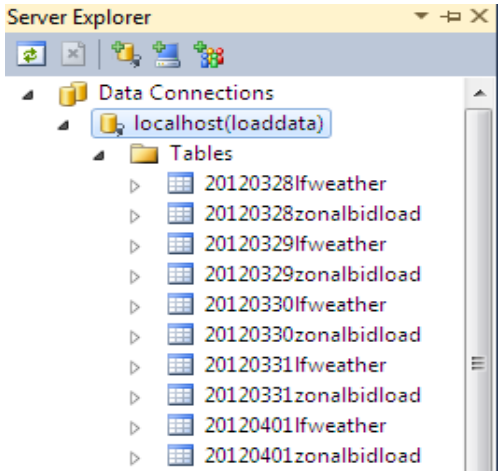

Figure 14. Tables in LoadData database screenshot.

In Figure 15, a partial content of the table 20120405zonalbidload is shown. This table is created from the Zonal Load Commitment file for April 5, 2012. This particular table has five different columns. The TimeStamp column contains the timestamp when the energy load was recorded. In this particular figure, the timestamps increase every hour.

20120405zonalbidlo...): Query(localhost)

TimeStamp	TimeZone	Name	PTID	EnergyLoad
4/5/2012 12:00:00 AM	EDT	CAPITL	61757	865
4/5/2012 12:00:00 AM	EDT	CENTRL	61754	1096
4/5/2012 12:00:00 AM	EDT	DUNWOD	61760	163
4/5/2012 12:00:00 AM	EDT	GENESE	61753	491
4/5/2012 12:00:00 AM	EDT	HUD VL	61758	617
4/5/2012 12:00:00 AM	EDT	LONGIL	61762	1355
4/5/2012 12:00:00 AM	EDT	MHK VL	61756	455
4/5/2012 12:00:00 AM	EDT	MILLWD	61759	156
4/5/2012 12:00:00 AM	EDT	N.Y.C.	61761	1758
4/5/2012 12:00:00 AM	EDT	NORTH	61755	96
4/5/2012 12:00:00 AM	EDT	WEST	61752	831
4/5/2012 1:00:00 AM	EDT	CAPITL	61757	793
4/5/2012 1:00:00 AM	EDT	CENTRL	61754	1053
4/5/2012 1:00:00 AM	EDT	DUNWOD	61760	155
4/5/2012 1:00:00 AM	EDT	GENESE	61753	472
4/5/2012 1:00:00 AM	EDT	HUD VL	61758	581
4/5/2012 1:00:00 AM	EDT	LONGIL	61762	1255
4/5/2012 1:00:00 AM	EDT	MHK VL	61756	424

25 of 264

Figure 15. Sample data in a table.

After successfully downloading the files onto a user's local machine, the next step is to upload data in the files into MySQL database tables. The use case in Figure 16 begins when a user clicks the "Upload To MySQL" button. When uploading data into database tables, new tables are created for each downloaded file. A table is named after the file from which it reads data on the NYISO's website, without the file extension. The Energy Data Analyzer System

figures out which databases in which to create tables. Finally, the data are uploaded into the newly created tables.

Name: Upload Selected Files to MySQL Database Tables

Description:
The user clicks the Upload To MySQL button to upload data from the downloaded files to the MySQL database tables.

Pre-Conditions:

- The user has selected files to download, and the tree view control to select files is disabled.
- Date selection is disabled, and the user has clicked the download button, successfully downloading the selected files.
- The progress bar shows completed status, and the label next to it says download complete.
- The Upload To MySQL button is visible.
- The table template exists.

Post-Conditions:

- All upload complete text is visible near the location where the Upload To MySQL button was.
- The Upload To MySQL button, progress bar, and download complete text are not visible.
- Data from files are uploaded into tables.

Basic Scenario:

1. The use case begins when the user clicks the Upload To MySQL button.
2. All upload complete text is visible.
3. The use case ends.

Alternate Scenario A:

A1. The user clicks the Upload To MySQL button.
A2. The system informs the user that the file could not be read.
A3. The user closes the error message and clicks the Refresh button.
A4. The use case ends.

Figure 16. Upload To MySQL use case.

3.5. Query

The Query tab in the Energy Data Analyzer System implements querying the database tables in the LoadData and PricingData databases. A user can select the dates and files to be queried, and based on these entries, the Energy Data Analyzer System identifies the tables in the database from which to query. If the files are not in the database yet, then the user is notified. The Energy Data Analyzer System provides a customized querying capability for a few file types. Under Load Data, built-in queries are available for the Zonal Load Commitment and the Actual Load Real-time. Under Pricing Data, built-in queries are available for the day-ahead market LBMP and the real-time market LBMP based on zones. Figure 17 shows the logic behind the Query tab implementation.

Initial Condition: Valid dates are selected; files selected for those dates may or may not be downloaded yet, and may or may not exist in the PricingData and LoadData databases.

Logic:

```
for each selected file
        identify the database to which it should belong
        check whether a database table for the file exists
        if exists, then
                show the available queries
        else
        display error message for the user, notifying which tables do not exist
end
if user query type, then
        enable/disable choosing criteria
        execute query
        output result in csv format
end
```

Figure 17. Query.

The Query tab allows users to query the tables in the LoadData and PricingData databases. Users can select the date and the type of data, and based on whether they are LoadData or PricingData, a dropdown box with a list of available table types for querying is populated. Then, step 3 lists the queries available for that table type. In the screenshot in Figure 18, the user has selected the query to find the average by zones for the Zonal Load Commitment on April 5, 2012. Step 4 lists the filters that can be applied to the user's query.

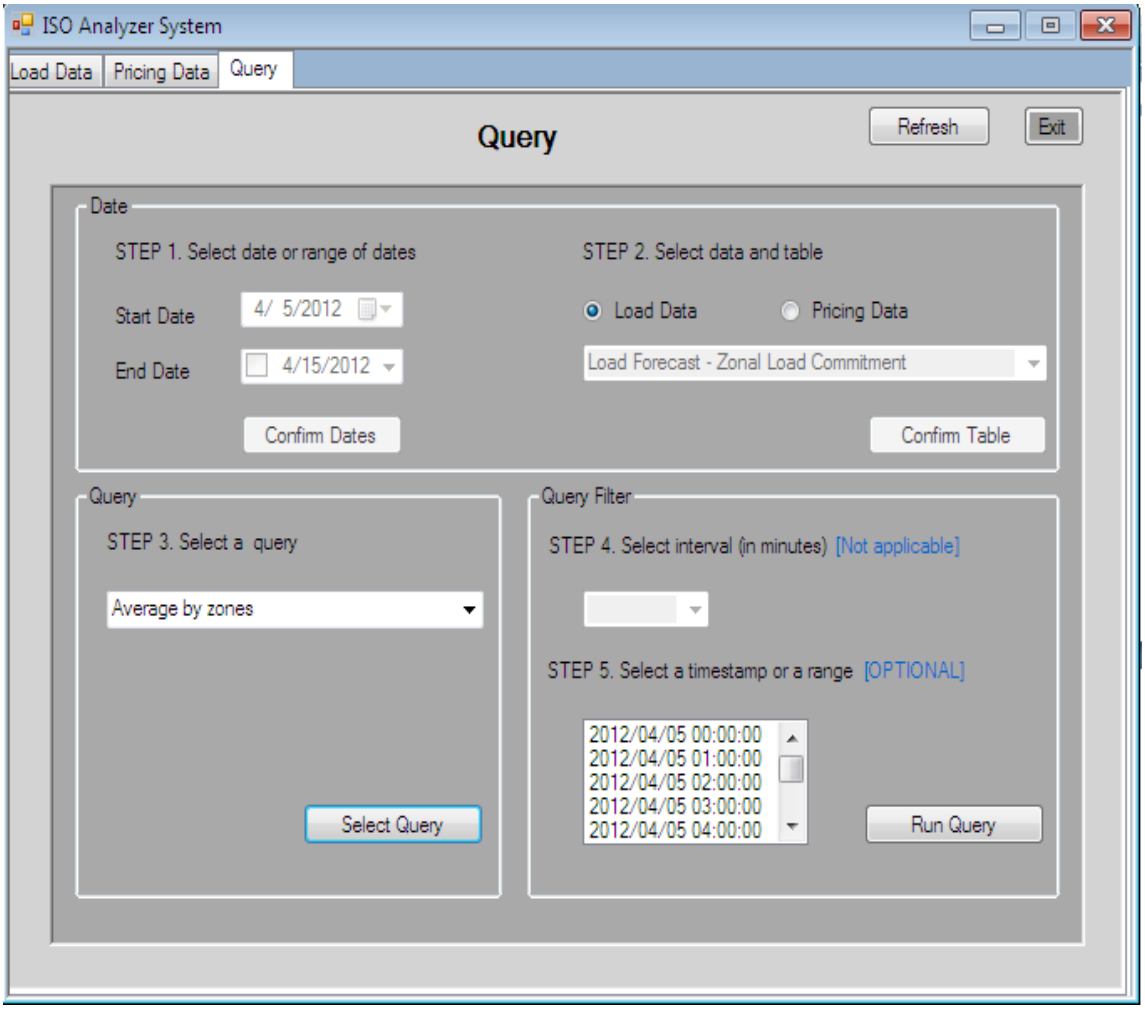

Figure 18. Energy Data Analyzer System screenshot 6.

3.6. Custom Queries

Dealing with large databases involving repeated human computer interaction, such as selecting some data, viewing them, or presenting them as desired, requires a user interface mechanism to access and manipulate the data for functionality and usability [19, 20]. The Energy Data Analyzer System provides an intuitive GUI to maximize both functionality and usability. Also, the custom queries are dynamically created, so the user does not have to write a new query each time. The Energy Data Analyzer System uses a dynamic query as an interactive technique, allowing users to input information from the GUI and generating queries at run time.

The Energy Data Analyzer System provides custom queries for the MySQL database tables created from Zonal Load Commitment, Actual Load Real-Time, Day-Ahead Market LBMP, and Day-Ahead Market LBMP data files. Queries available for Zonal Load Commitment and Day-Ahead Market LBMP data files are average by zones, view all records, and delete selected tables for the dates. The queries available for Actual Load Real-Time and Real-Time Market LBMP data files include two additional queries: zonal average by timestamp interval query and timestamp interval average by zones query.

The average by zones query calculates the average energy load or average LBMP. The view all records query returns all the records from the queried table(s). The delete selected tables for the dates query deletes the tables for the selected dates. The zonal average by timestamp interval allows users to select time intervals and to view data based on different time intervals, such as listing the data for five-minute intervals or ten-minute intervals. The intervals provided in the GUI are multiples of five. Also, the timestamp interval average by zones query allows a user to select the time intervals and generates the result grouped by zones.

Query optimization is important in decision-support applications when queries involve grouping and aggregation. Using SQL views rather than accessing the database relation speeds up query processing because some of the necessary processing would have already been completed while creating the views [21, 22, 23]. The Energy Data Analyzer System enables users to run powerful queries involving aggregation and grouping, such as calculating load averages and pricing averages for arbitrary time intervals for various zones. SQL views are used as part of the query to accomplish time-interval average calculations.

The result of a query is a CSV file saved in the personal folder of the user. If the user is using Windows, then it is saved in the My Documents folder. Because a file name cannot be overwritten using a query language, a time sequence has been added to the result file name so that the new file is named differently than any existing files. The result can be further manipulated using the Excel application to filter with certain criteria, to sort, or to plot graphs and create a report.

Figure 19 shows the basic and alternative scenarios for query operation. The use case for a query begins when a user wants to run a query and selects the Query tab. The user selects a date or a date range as well as pricing or load data, and then selects the table(s) to be queried. This process creates names of the tables to be queried by concatenating the dates and the table names. The Energy Data Analyzer System then checks to see whether there are any tables in the database that match names of the tables to be queried. If a table to be queried is missing from the database, the user is notified by an error message display. The next course of action would be downloading that particular file and uploading the contents to the MySQL database for querying.

Name: Query Database Tables

Description:
The user wants to run a query.

Pre-Conditions:

- The PricingData and LoadData databases exist.

Post-Conditions:

- The query result is saved in a CSV file in a user's personal folder. In Windows, the result is saved in the My Documents folder.

Basic Scenario:

1. The use case begins when a user wants to run a query and clicks the Query tab.
2. The user selects dates and clicks the Confirm Date button.
3. The user chooses Load Data.
4. The user selects a table name from the dropdown box.
5. The user selects one of the queries available for the table.
6. The user selects criteria and clicks the Run Query button.
7. The use case ends when a query result is saved as a .csv file.

Alternate Scenario A:

A1. The user selects a table for a date, but the database does not have the table.
A2. The system informs the user that files have to be downloaded to create the table.
A3. The use case ends.

Alternate Scenario B:

B1. The user selects dates and clicks the Confirm Date button.
B2. The user chooses Load Data and selects a table name from a dropdown box.
B3. The user decides to select a different date.
B4. The user clicks the Refresh button.
B5. The use case begins from Basic Scenario step 1.

Figure 19. Query use case.

3.7. Energy Data Analyzer System Class Diagram

Figure 20 is the static structure of the Energy Data Analyzer System. We will briefly go over the class diagram after presenting the figure.

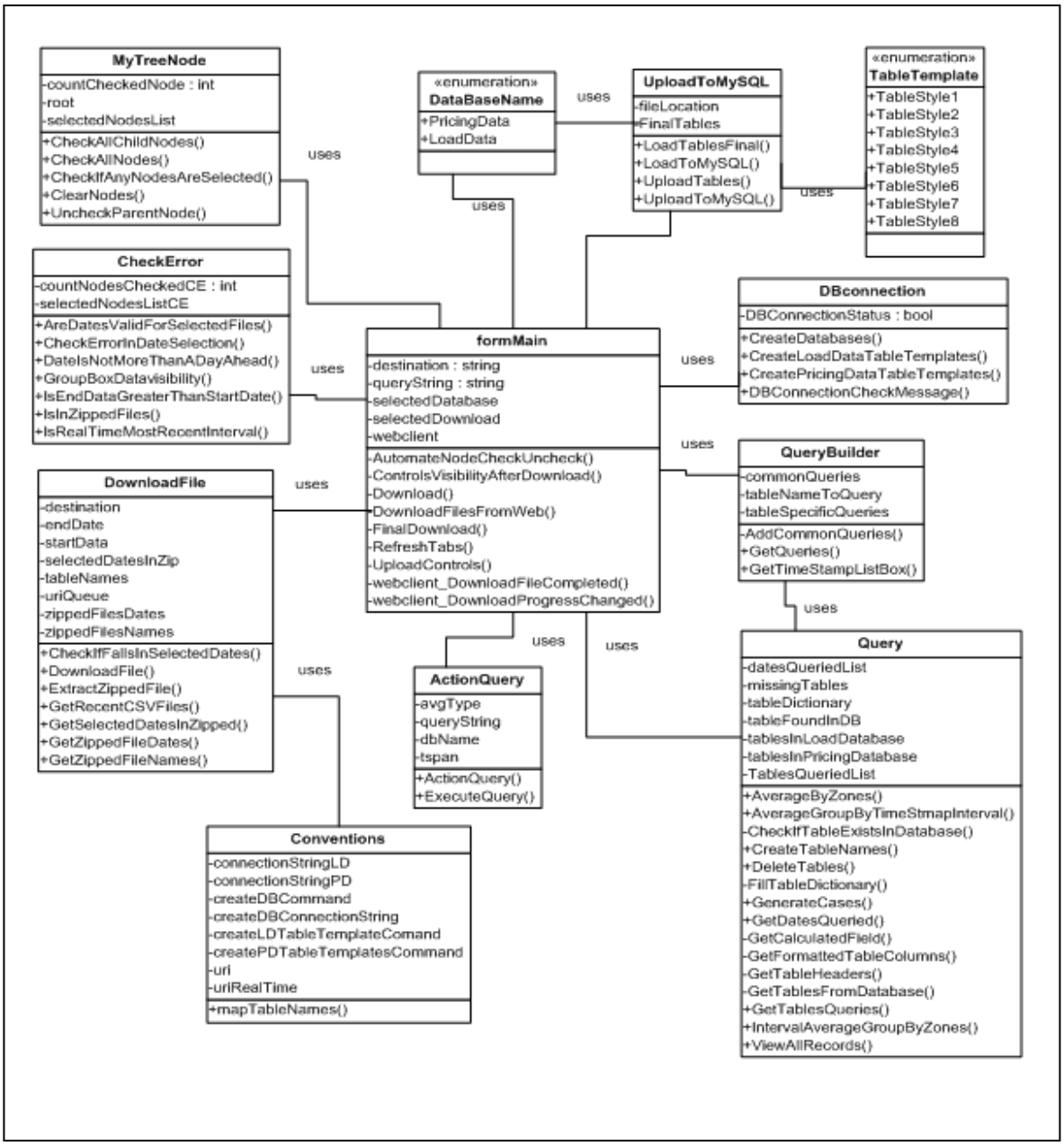

Figure 20. Energy Data Analyzer System class diagram.

Figure 20 shows the classes that make up the Energy Data Analyzer System. Each class has its own set of fields and methods. For readability and space, the getters and setters methods are not included in the diagram. In the figure, formMain class contains event handlers which are called when a user interacts with the system. For example, when the user clicks the Download button, the event handler for the Download button is called. The tasks assigned to the event handlers are done by calling various methods in the classes provided. For example, MyTreeNode class methods are called to automatically check on the boxes next to all the child nodes if their parent node is checked in the tree view.

4. DEMONSTRATION

This chapter illustrates how the Energy Data Analyzer System allows a file for a certain date to be downloaded from the NYISO website and how the data from the file are uploaded into a MySQL database table for querying. When the Energy Data Analyzer System program is run, the LoadData and PricingData databases as well as the table templates, tablestyle1 through tablestyle8, are created if they did not already exist. Let us suppose a user wants to download the Real-Time Actual Load data file about load data for April 5, 2012. After making the date and the file selections, the user clicks the Download button to download the file. Figure 21 shows the file and the date selections as well as the download-completed status.

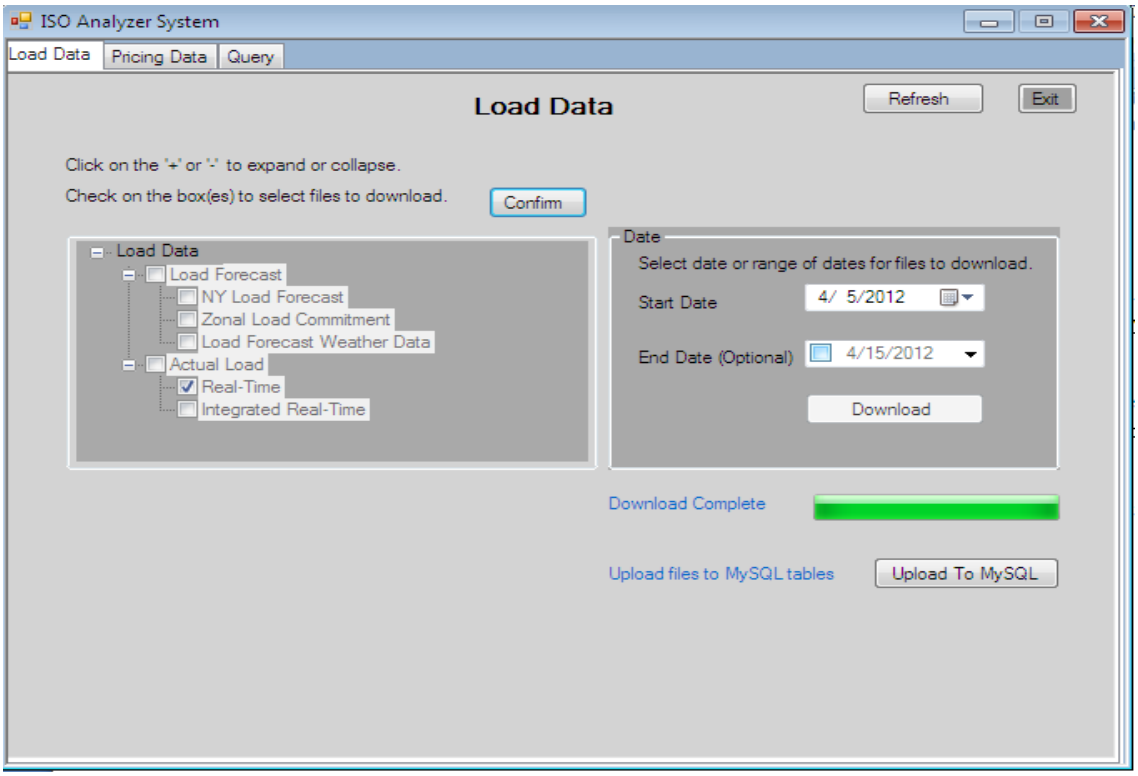

Figure 21. Energy Data Analyzer System screenshot 7.

After clicking the Upload To MySQL button, the data from the downloaded file are uploaded to a new MySQL table. Figure 22 shows a partial screenshot of the new table as seen using Visual Studio 2010.

20120405pal (loaddata): Query(localhost)

TimeStamp	TimeZone	Name	PTID	EnergyLoad
4/5/2012 12:00:00 AM	EDT	CAPITL	61757	1086.7
4/5/2012 12:00:00 AM	EDT	CENTRL	61754	1503.5
4/5/2012 12:00:00 AM	EDT	DUNWOD	61760	526.5
4/5/2012 12:00:00 AM	EDT	GENESE	61753	923.2
4/5/2012 12:00:00 AM	EDT	HUD VL	61758	862.9
4/5/2012 12:00:00 AM	EDT	LONGIL	61762	1893.3
4/5/2012 12:00:00 AM	EDT	MHK VL	61756	786
4/5/2012 12:00:00 AM	EDT	MILLWD	61759	262.7
4/5/2012 12:00:00 AM	EDT	N.Y.C.	61761	4782.9
4/5/2012 12:00:00 AM	EDT	NORTH	61755	705.3
4/5/2012 12:00:00 AM	EDT	WEST	61752	1592.1
4/5/2012 12:05:00 AM	EDT	CAPITL	61757	1079.2
4/5/2012 12:05:00 AM	EDT	CENTRL	61754	1514.2
4/5/2012 12:05:00 AM	EDT	DUNWOD	61760	519.3
4/5/2012 12:05:00 AM	EDT	GENESE	61753	915.8
4/5/2012 12:05:00 AM	EDT	HUD VL	61758	867.3
4/5/2012 12:05:00 AM	EDT	LONGIL	61762	1853.5
4/5/2012 12:05:00 AM	EDT	MHK VL	61756	782.7
4/5/2012 12:05:00 AM	EDT	MILLWD	61759	254.1
4/5/2012 12:05:00 AM	EDT	N.Y.C.	61761	4750.6
4/5/2012 12:05:00 AM	EDT	NORTH	61755	692.7
4/5/2012 12:05:00 AM	EDT	WEST	61752	1584.9
4/5/2012 12:10:00 AM	EDT	CAPITL	61757	1083.4
4/5/2012 12:10:00 AM	EDT	CENTRL	61754	1471.5
4/5/2012 12:10:00 AM	EDT	DUNWOD	61760	508.8
4/5/2012 12:10:00 AM	EDT	GENESE	61753	908

31 of 3179

Figure 22. Sample table data after MySQL upload.

Suppose that the Energy Data Analyzer System has just downloaded a file to be queried and that the data from the file are already uploaded in a table called 20120405pal. The Query tab is demonstrated in Figure 23 with a user case.

Name: Zonal Average by Timestamp Interval

Description:
The user wants to Zonal average by timestamp interval query for Load Actual–Real-Time data for April 5, 2012.

Pre-Conditions:

- The PricingData and LoadData databases exist. Data from the downloaded .CSV file have been uploaded into table 20120405pal.

Post-Conditions:

- The query result is saved in a .csv file in the user's personal folder. In Windows, the result is saved in the My Documents folder.

Basic Scenario:

1. This use case begins when a user wants to run the "Zonal average by timestamp Interval" query.
2. In STEP 1, April 5, 2012, is selected, and the Confirm Dates button is clicked.
3. In STEP 2, the Load Data radio button is checked, and Load Actual – Real-Time table is selected from the dropdown box.
4. The user clicks the Confirm Table button.
5. The user clicks on Zonal Average By Timestamp Interval in the Query box and clicks Select Query.
6. When clicking Select Query, query filters that can be applied to the query before running it are shown.
7. The user moves to STEP 4, Select interval (in minutes), and selects 10 from the dropdown box in the Query Filter group box.
8. The user clicks Run Query.
9. A success message is displayed in a message box.
10. A .csv result file is created in the My Documents folder; the use case ends.

Figure 23. Zonal average by timestamp interval query basic use case.

The result of the query is in .csv format which can be easily opened in Excel. Figure 24 illustrates the result in the form of an Excel file showing data in a tabular format. It shows the average of all zones every 10 minutes. The average energy load for the zones at 10 minutes past midnight is 1351.79 units.

	A	B
1	Average	Timestamp
2	1351.79	4/5/2012 0:00
3	1333	4/5/2012 0:10
4	1323.97	4/5/2012 0:20
5	1311.39	4/5/2012 0:30
6	1300.1	4/5/2012 0:40
7	1288.81	4/5/2012 0:50
8	1282.1	4/5/2012 1:00
9	1272.73	4/5/2012 1:10
10	1264.33	4/5/2012 1:20
11	1257.29	4/5/2012 1:30
12	1246.73	4/5/2012 1:40
13	1245.59	4/5/2012 1:50
14	1237.29	4/5/2012 2:00
15	1235.49	4/5/2012 2:10
16	1231.45	4/5/2012 2:20
17	1223.59	4/5/2012 2:30
18	1218.6	4/5/2012 2:40
19	1216.49	4/5/2012 2:50
20	1211.09	4/5/2012 3:00
21	1212.75	4/5/2012 3:10
22	1212.81	4/5/2012 3:20
23	1217.02	4/5/2012 3:30
24	1215	4/5/2012 3:40
25	1217.77	4/5/2012 3:50
26	1220.53	4/5/2012 4:00
27	1222.45	4/5/2012 4:10

Figure 24. Zonal average by timestamp interval query result.

Similarly, if another query, Timestamp interval average by zones, was selected, the resulting file would have data for every 10-minute interval grouped by zones. Figure 25 shows the query's result which is opened in Microsoft Excel. The .csv output file can be easily opened using Excel. With Excel, one can apply additional filters. Here, additional filters to show 10-minute averages from 4 am and 4:10 am have been added, in Excel, to the ISO Analyzer's result file.

	A	B	C
1	Zones	Averag	Timestamp
26	CAPITL	1016.8	4/5/2012 4:00
27	CAPITL	1029.6	4/5/2012 4:10
170	CENTRL	1468.5	4/5/2012 4:00
171	CENTRL	1424.9	4/5/2012 4:10
314	DUNWOD	443.05	4/5/2012 4:00
315	DUNWOD	445.5	4/5/2012 4:10
458	GENESE	846.4	4/5/2012 4:00
459	GENESE	861.05	4/5/2012 4:10
602	HUD VL	768.5	4/5/2012 4:00
603	HUD VL	773.9	4/5/2012 4:10
746	LONGIL	1633.05	4/5/2012 4:00
747	LONGIL	1647.05	4/5/2012 4:10
890	MHK VL	719.45	4/5/2012 4:00
891	MHK VL	721.3	4/5/2012 4:10
1034	MILLWD	248.3	4/5/2012 4:00
1035	MILLWD	248.8	4/5/2012 4:10
1178	N.Y.C.	4069.45	4/5/2012 4:00
1179	N.Y.C.	4076	4/5/2012 4:10
1322	NORTH	696.55	4/5/2012 4:00
1323	NORTH	698.35	4/5/2012 4:10
1466	WEST	1515.75	4/5/2012 4:00
1467	WEST	1520.45	4/5/2012 4:10

Figure 25. Timestamp interval average by zones.

Here is an alternative use case scenario for Figure 23. When a user decides to query a database, the tables the query needs may not exist in the database. Then, a user has to download the related files from the NYISO website and upload the data into the tables. An appropriate error message or instructions need to be given if the user downloaded the files but did not upload the data into tables or if the user went directly to the Query tab to run the queries without downloading the needed files. Figure 26 describes such a use case.

Name: Zonal Average by Timestamp Interval Query Exception Use Case

Description:
The user wants to run a Zonal average by timestamp interval query for Load Actual–Real-Time data for April 6, 2012, through April 8, 2012.

Pre-Conditions:

- The PricingData and LoadData databases exist. The needed CSV files have not been downloaded or uploaded into MySQL database tables.

Post-Conditions:

- An error message is displayed, and the user cannot proceed to STEP 3 in the Query tab.

Exception Scenario:

1. This use case begins when the user wants to run a "Zonal average by timestamp interval" query.
2. In STEP 1, April 6, 2012, through April 8, 2012, is selected, and the Confirm Dates button is clicked.
3. In STEP 2, the Load Data radio button is checked, and the Load Actual–Real-Time table is selected from the dropdown box.
4. When clicking the Confirm Table button, an error message saying the tables were not found is displayed.
5. The user clicks the Refresh button. The exception use case ends.

Figure 26. Zonal average by timestamp interval query exception use case.

In Figure 27, a user wants to run a query on the tables that are supposed to hold data for the Real-Time Actual Load from April 6, 2012, through April 8, 2012. However, the related files have neither been downloaded from NYISO nor uploaded into the MySQL tables. In this case, an error message displays the missing tables. The user has to correct the error before proceeding to the next step. Until then, available queries for the table type are not visible to the user.

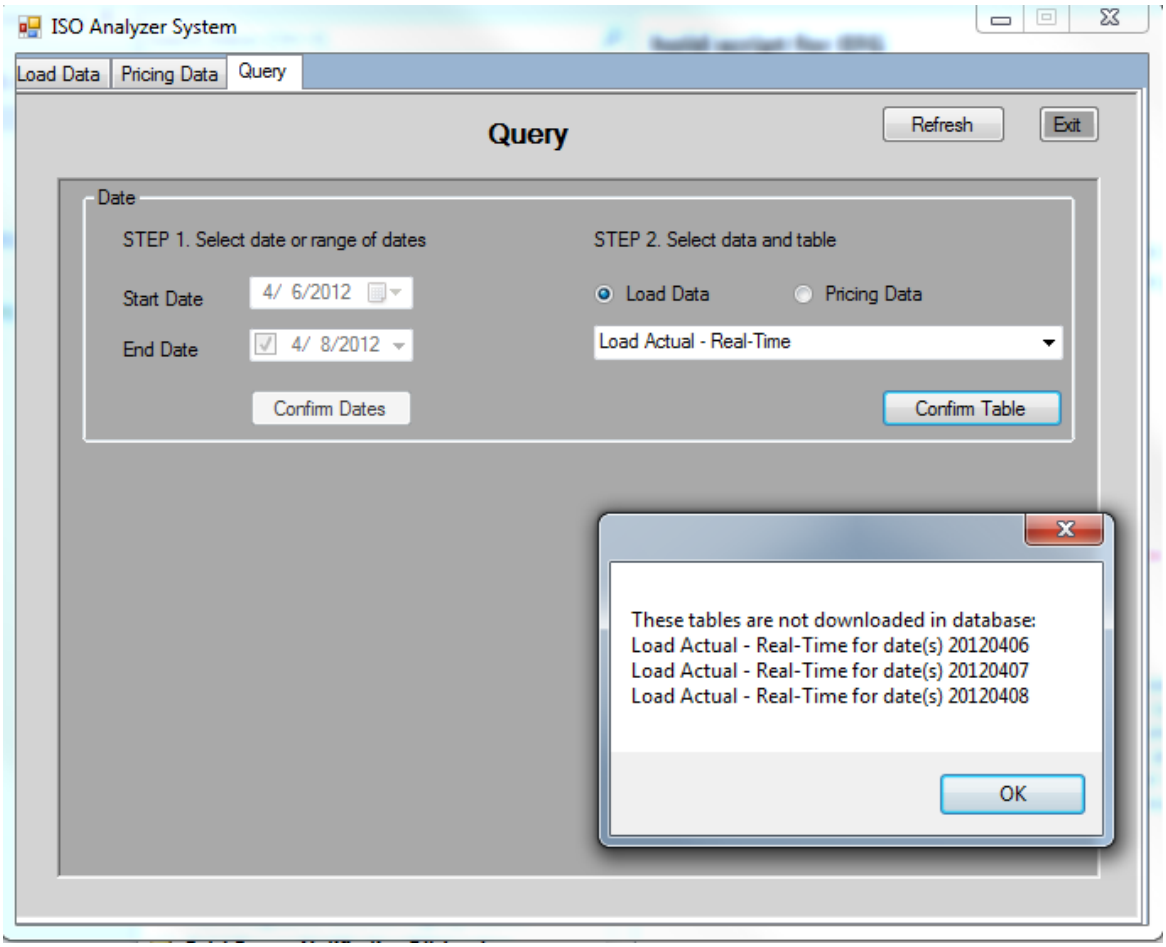

Figure 27. Energy Data Analyzer System screenshot 8.

5. CONCLUSION AND FUTURE WORK

The Energy Data Analyzer System meets the requirements presented in chapter 1, Introduction, by empowering the user with the ability to retrieve the energy data files from the NYISO's website, download them in the user's local folder, and upload the data into MySQL database tables for querying. The Energy Data Analyzer System also provides powerful dynamic custom queries to calculate the average energy load and pricing during various time intervals for the energy zones. Users are able to select the time intervals, which are in multiples of five; the maximum being sixty-minute interval. The results of custom queries from the Energy Data Analyzer System are in the CSV file format. These CSV files can be easily opened using Excel; the files can be further filtered and manipulated to sort or to create graphs and charts.

Thus, the Energy Data Analyzer System allows an efficient analysis of load data and pricing dataover a period of time by storing the data as database tables which can be queried to get the desired information. The Energy Data Analyzer System automates the repetitive tasks of downloading the energy data files from the NYISO's website and creating tables for them, making it easier for the user to visualize the trends in electricity markets. At this time, only certain files are implemented to be queried using the provided GUI, however, other load data and pricing data files can be downloaded and uploaded into MySQL tables. Users can still run queries by installing the MySQL command Line Client. Also, custom queries for other load and pricing data as well as additional criteria to add flexibility to the queries could be studied in the future to expand the system.

REFERENCES

1. New York Independent System Operator, http://www.nyiso.com/public/markets_operations/index.jsp, accessed April 2012.

2. Load Data and Pricing Data Files, http://mis.nyiso.com/public/, accessed April 2012.

3. C# Language Specification, http://msdn.microsoft.com/en-us/library/aa292164%28v=vs.71%29.aspx, accessed April 2012.

4. Christian Nagel, Bill Evjen, Jay Glynn, Karli Watson, and Morgan Skinner, Professional C# 2008, Wiley Publishing, Inc. Indianapolis, IN, United States. 2008.

5. Introducing Visual Studio.Net, http://msdn.microsoft.com/en-us/library/fx6bk1f4%28v=vs.71%29.aspx, accessed April 2012.

6. Barbara Doyle, Microsoft Visual C#.NET Programming: From Problem Analysis to Program Design, Course Technology Press Boston, MA, United States. 2004.

7. Base Class Library, http://msdn.microsoft.com/en-us/netframework/aa569603.aspx, accessed April 2012.

8. .NET Framework, http://msdn.microsoft.com/en-us/netframework/, accessed April 2012.

9. Visual Studio 2010 Premium, http://www.microsoft.com/visualstudio/en-us/products/2010-editions/premium/overview, accessed April 2012.

10. MySQL, http://www.mysql.com/why-mysql/, accessed April 2012.

11. SQL, http://www.w3schools.com/sql/default.asp, accessed April 2012.

12. MySQL Connector Net, http://dev.mysql.com/doc/refman/5.0/en/connector-net.html, accessed April 2012.

13. Locational Based Margin Price Zones http://www.eei.org/meetings/MeetingDocuments/NYISOnelson.ppt, accessed April 2012.

14. Locational Based Margin Price, http://www.nyiso.com/public/about_nyiso/understanding_the_markets/cost_of_electricity/index.jsp, accessed April 2012.

15. NYISO Energy Markets, http://www.nyiso.com/public/about_nyiso/understanding_the_markets/energy_market/index.jsp, accessed April 2012.

16. WebClient Class, http://msdn.microsoft.com/en-us/library/system.net.webclient%28v=vs.80%29.aspx, accessed April 2012.

17. How to: Use a Background Worker, http://msdn.microsoft.com/en-us/library/cc221403%28v=vs.95%29.aspx, accessed April 2012.

18. DotNetZip Library, http://dotnetzip.codeplex.com/, accessed April 2012.

19. Jade Goldstein and Steven F. Roth, Using Aggregation and Dynamic Queries for Exploring Large Data Sets. In Proceedings of CHI'94 Human Factors in Computing Systems, April 24-28, 1994, Boston, MA, p.23-29.

20. Christopher Ahlberg, Christopher Williamson, and Ben Shneiderman, Dynamic Queries for Information Exploration: An Implementation and Evaluation. In Proceedings of CHI'92 Human Factors in Computing, May 3-7, 1992, Monterey, CA, p.619-626.

21. Alon Y. Halevy, Answering Queries Using Views: A Survey. The International Journal on Very Large Data Bases, v.10 n.4, December 2001, p.270-294.

22. Foto Afrati, Rada Chirkova, Manolis Gergatsoulis, and Vassia Pavlaki, Finding Equivalent Rewritings in the Presence of Arithmetic Comparisons. In Proceedings of the 10th International Conference on Advances in Database Technology, p.1-2, 2006.

23. Foto Aftrati, Chen Li, Prasenjit Mitra, Rewriting Queries Using Views in the Presence of Arithmetic Comparisions, Theoretical Computer Science, v.368 n.1-2, December, 2006, p.88-123.