MINING WEATHER DATA

A WEB APPLICATION FOR CALIFORNIA SMART GRID CENTER

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

Nikitha Thouta

SPRING 2015
MINING WEATHER DATA
A WEB APPLICATION FOR CALIFORNIA SMART GRID CENTER

A Project

by

Nikitha Thouta

Approved by:

__________________________________, Committee Chair
Dr. Meiliu Lu

__________________________________, Second Reader
Mr. Russ Tatro

__________________________________
Date
Student: Nikitha Thouta

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
Dr. Jinsong Ouyang

Department of Computer Science
Abstract

of

MINING WEATHER DATA

A WEB APPLICATION FOR CALIFORNIA SMART GRID CENTER

by

Nikitha Thouta

"Mining weather data" is a web application specifically developed for the California Smart Grid Center at the California State University, Sacramento. The main objective of this web application is to help the researchers at Smart Grid Research Center to visualize the historic weather data in the form of graphs. The main purpose of this project is to overcome the tedious process of reading huge amounts of raw data generated from the censors and analyze it manually.

The application allows the user to choose a time period and weather parameter for which the data needs to be visualized. The application extracts day-to-day weather prediction from historical data using Online Analytical Process (OLAP) technique. OLAP is a technique used to analyze the data in different dimensions. The application then uses JFree Charts to display the filtered data in graphical form. The data is displayed in the form of line graph where X-axis is the time range given by the user and Y-axis is the weather parameter data. The JFree charts are then converted to .png files and sent to the application to display over the server.

_______________________, Committee Chair
Dr. Meiliu Lu

_______________________
Date
ACKNOWLEDGMENTS

I would like to thank Dr. Meiliu Lu for her essential feedback throughout the development of this project. Dr. Meiliu Lu was extremely supportive and understanding which provided me great confidence in completing my project as expected. She has been very cooperative during course of my project especially considering me working on my project remotely.

I would like to thank Mr. Russ Tatro for giving me an excellent opportunity for being a part of California Smart Grid Center Research project. His support and encouragement has helped me in completing the project as expected. Also I would like to specially thank Mr. Tatro for taking time to review my report and providing me precious feedback.

I would also like to thank Department of Computer Science and Office of Global Education for their enormous help in completing my Masters at California State University, Sacramento.

Last but not the least, I am very thankful to my family for believing in me and being a constant support all throughout my Masters degree study.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td></td>
<td>vi</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td></td>
<td>xi</td>
</tr>
<tr>
<td>CHAPTER</td>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1.1 Overview</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1.2 Background</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1.3 Scope</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1.3.1 In-Scope</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>1.3.2 Out-Scope</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>1.3.3 Future Work</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>1.4 Technologies Used</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>1.5 Tools Used</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>1.6 Libraries Used</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>2. APPLICATION ARCHITECTURE AND DESIGN</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>2.1 Application Architecture</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>2.1.1 Presentation Layer</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>2.1.2 Application Layer</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>2.1.3 Data Layer</td>
<td>11</td>
</tr>
</tbody>
</table>
3. ONLINE ANALYTICAL PROCESSING .......................................................... 13

3.1 OLAP Types ......................................................................................... 13
3.1.1 Multidimensional OLAP ................................................................. 14
3.1.2 Relational OLAP ............................................................................. 14
3.1.3 Hybrid OLAP ................................................................................ 17

3.2 OLAP Operations .................................................................................. 17
3.2.1 Roll Up Operation ........................................................................... 18
3.2.2 Drill Down Operation .................................................................... 19
3.2.3 Slice Operation .............................................................................. 20
3.2.4 Dice Operation ............................................................................... 21

4. IMPLEMENTATION .................................................................................. 22

4.1 Data Mart Implementation ................................................................... 22
4.1.1 Sample Queries Used .................................................................... 24

4.2 User Interface Implementation ............................................................. 25

4.3 Implementation of OLAP in Mining Weather Data Application .......... 29

4.4 JFreeCharts Implementation ............................................................... 31

4.5 Application Flow .................................................................................. 35
4.5.1 Application flow for User ............................................................... 35
4.5.2 Application flow for System ............................................................ 36
4.5.3 Application flow for Range Selection ............................................ 37
4.5.4 Application flow for Date Selection .............................................. 38
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1. List of Weather Parameters</td>
<td>3</td>
</tr>
<tr>
<td>Table 2. Tables Used</td>
<td>22</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1. Architectural Diagram</td>
<td>10</td>
</tr>
<tr>
<td>Figure 2. Data Layer Schema</td>
<td>12</td>
</tr>
<tr>
<td>Figure 3. Sample Query</td>
<td>13</td>
</tr>
<tr>
<td>Figure 4. Two tier architecture for data processing</td>
<td>15</td>
</tr>
<tr>
<td>Figure 5. Three tier architecture for data processing</td>
<td>16</td>
</tr>
<tr>
<td>Figure 6. Roll Up Operation</td>
<td>18</td>
</tr>
<tr>
<td>Figure 7. Drill Down Operation</td>
<td>19</td>
</tr>
<tr>
<td>Figure 8. Slice Operation</td>
<td>20</td>
</tr>
<tr>
<td>Figure 9. Dice Operation</td>
<td>21</td>
</tr>
<tr>
<td>Figure 10. Star Schema for Mining Weather Data</td>
<td>23</td>
</tr>
<tr>
<td>Figure 11. Connect Statement</td>
<td>24</td>
</tr>
<tr>
<td>Figure 12. Create Statement</td>
<td>24</td>
</tr>
<tr>
<td>Figure 13. Select Statement</td>
<td>25</td>
</tr>
<tr>
<td>Figure 14. Home Screen</td>
<td>26</td>
</tr>
<tr>
<td>Figure 15. Graph for Range Selection</td>
<td>27</td>
</tr>
<tr>
<td>Figure 16. Graph for Date Selection</td>
<td>28</td>
</tr>
<tr>
<td>Figure 17. Process Flow for OLAP Technique</td>
<td>29</td>
</tr>
<tr>
<td>Figure 18. Application flow of Jfree Chart</td>
<td>32</td>
</tr>
<tr>
<td>Figure 19. Weather Chart</td>
<td>34</td>
</tr>
</tbody>
</table>
Figure 20. User Application flow ................................................................. 35
Figure 21. System Application flow ............................................................... 36
Figure 22. Range Selection Application flow ................................................... 37
Figure 23. Date Selection Application flow ...................................................... 38
Figure 24. Folder Location ............................................................................. 39
Figure 25. Start Tomcat Server ....................................................................... 40
Figure 26. Sample localhost address ............................................................... 41
Figure 27. Successful Web page ..................................................................... 41
Figure 28. Range Selection Screen ................................................................. 42
Figure 29. Date Selection Screen .................................................................. 42
Figure 30. Weather Parameter Selection Screen .............................................. 42
Figure 31. Generate Graph ............................................................................ 43
Figure 32. Sample Graph .............................................................................. 43
Chapter 1

INTRODUCTION

1.1 Overview
Weather forecasting is a common term which we use in our day-to-day life. It involves prediction of what the weather will likely to be in future. In the past decade, modern technologies have invented several computer models to provide an accurate weather prediction. Weather forecasting is something that needs to be performed every millisecond, and the data that is retrieved is very huge. Due to the increasing availability of the weather data, it is important to find an effective and accurate procedure to analyze and extract knowledge from this huge data. The purpose of this project is to analyze the weather data and generate graphs dynamically for a user provided data which includes time range and weather parameter.

1.2 Background
The California SmartGrid Center (CSGC) at California State University, Sacramento mainly focuses to pull our nation away from relying on traditional, non-renewable energy sources while encouraging the use of renewable energy sources [1]. The CSGC is performing several research projects in order to improve the electricity delivery systems of the state's power grid. One of their main targets is Home Area Networks (HAN) which provides power supply for the residential areas.
HAN mainly focuses on optimizing the usage of the power generated by a power grid based on the weather conditions. For example, if the solar input is good enough to use a solar power system, the SmartGrid automatically switches from powergrid to solar power. In order to optimize the power supply, the research focuses on predicting the weather based on past weather patterns. The project mainly concentrates on providing the patterns in the form of graphs to the research team.

The researchers at California State University, Sacramento (CSUS) has collected huge weather data from past couple years, which has been calculated from multiple sensors that are located in different places in Riverside Hall at CSUS. Each sensor calculates different weather parameter such as temperature, wind speed, humidity etc and stores it in the form of .csv files. Our application extracts data from this .csv files, analyzes data using data mining techniques and generates graphs for the end user.

1.3 Scope

This section describes the scope of the application. As this application is a part of SmartGrid research which is very huge, the scope can be categorized into three different parts:

- In-Scope: Describes the features that are implemented in the current application.
- Out-Scope: Describes the features that need to be implemented in near future.
- Future Work: Describes the feature that can be extended to the current application.
1.3.1 In-Scope
The current features in this project include user interface where user can choose parameters like start date, end date and weather parameter dynamically. The application extracts the data from .csv file, performs data mining operations and generates graph. Currently the application uses one full year data where each record generated every second of the day. Each record generated has a unique RecordId, timestamp and sixteen weather parameters that are listed in Table 1:

<table>
<thead>
<tr>
<th>Global_Avg</th>
<th>WS_ms_Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct_Avg</td>
<td>Rain_mm_Tot</td>
</tr>
<tr>
<td>Diffuse_Avg</td>
<td>DNIBird_Avg</td>
</tr>
<tr>
<td>AirTemp_C_Avg</td>
<td>ZenDeg_Avg</td>
</tr>
<tr>
<td>RH_Avg</td>
<td>Global_Raw_Avg</td>
</tr>
<tr>
<td>Press_Avg</td>
<td>Direct_Raw_Avg</td>
</tr>
<tr>
<td>WS_ms_WVC(1)</td>
<td>Diffuse_Raw_Avg</td>
</tr>
<tr>
<td>WS_ms_WVC(2)</td>
<td>PSP_Avg</td>
</tr>
</tbody>
</table>

*Table 1: List of Weather Parameters*

1.3.2 Out-Scope
The application only supports manual Extract, Transform and Load (ETL) process. It is not automated in our current application due to time constraint and prioritizing other features over automation. The efficiency of the application can be improved by automatically updating the data.
1.3.3 Future Work

The following properties can make up the scope for future work of the application:

1. The application can be extended to generate graphs for all the weather parameters in a single click.
2. The data to the application can be retrieved from the server dynamically.
3. The application can be integrated with the California Smart Grid Website.
4. The application can be extended to use other type of data files like database file, excel file etc.
5. The application can be automated much more where it generates graph, applies computer models and predicts future weather.
6. The application can have a top layer security where it only allows access to the authorized users.

1.4 Technologies Used

Javascript: JavaScript is a dynamic programming language that is most commonly used as part of web browsers, whose implementations allow client-side scripts to interact with the user, control the browser, and alter the document content that is displayed [2]. It is also used in the development of games and desktop/mobile applications.
JavaScript is classified as a prototype-based scripting language with dynamic typing and first-class functions. It supports object-oriented, imperative, and functional programming styles. The syntax of JavaScript is actually derived from C, while the semantics and design are influenced by the Self and Scheme programming languages.

JavaScript is also used in non web-based applications, such as PDF documents and desktop widgets. Newer and faster JavaScript virtual machines (VMs) and platforms built upon them have also increased the popularity of JavaScript for server-side web applications. On the client side, JavaScript has been traditionally implemented as an interpreted language, but more recent browsers perform just-in-time compilation.

JavaScript has been standardized in the ECMAScript language specification.

**Hyper Text Markup Language (HTML):** Hyper Text Markup Language, which is widely known as html is a markup language that is used for web designing [3]. It was developed by physicist Tim Berners-Lee in the year 1980. It is simple to learn the language, yet it is quite powerful in terms of what can be created using it. HTML code ensures proper formatting of text and images displayed on a webpage. Most browsers are designed to read html and compile them into viewable, audible & interactive webpages. HTML coding consists of several components such as tags, elements & data types. Once HTML files are composed, they are transmitted to the worldwide web using Hypertext Transfer Protocol.
**Cascading Style Sheet (CSS):** Cascading Style Sheets (CSS) is a style sheet language used for describing the look and formatting of a document written in a markup language [4]. While most often the CSS is used to change the style of web pages and user interfaces written in HTML and XHTML, the language can be applied to any kind of XML document, including plain XML, SVG and XUL. Along with HTML and JavaScript, CSS is a cornerstone technology used by most websites to create visually engaging webpages, user interfaces for web applications, and user interfaces for many mobile applications.

CSS is designed primarily to enable the separation of document content from document presentation, including elements such as the layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple HTML pages to share formatting, and reduce complexity and repetition in the structural content. CSS makes it possible to separate presentation instructions from the HTML content in a separate file or style section of the HTML file. For each matching HTML element, it provides a list of formatting instructions.

This separation of formatting and content makes it possible to present the same markup page in different styles for different rendering methods. It can also be used to display the web page differently depending on the screen size or device on which it is being viewed.
1.5 Tools Used

**NetBeans:** NetBeans is an integrated development environment written in Java. It’s a software platform which allows applications to be developed from a set of modules. Most of the applications developed on NetBeans platform can be enhanced or modified by third party developers. NetBeans is intended for development in Java, but it is also compatible with C/C++, PHP and HTML5. NetBeans IDE can be installed on all operating systems that support Java, from Windows to Linux to Mac OS systems [5]. Applications that are developed using NetBeans IDE work across several OS systems without having to change the code because they are written in Java.

**Apache Tomcat:** Apache Tomcat is an open-source web server and servlet container developed by the Apache Software Foundation (ASF). Tomcat implements several Java EE specifications including Java Servlet, JavaServer Pages (JSP), Java EL, and WebSocket, and provides a pure Java" HTTP web server environment for Java code to run in [6].

Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation, released under the Apache License 2.0 license, and is open-source software.
1.6 Libraries Used

**JCommon library:** JCommon is a Java class library that is used by JFreeChart, Pentaho Reporting and a few other projects. The library contains miscellaneous classes that support:

- configuration and dependency management code
- text utilities
- user interface classes for displaying information about applications
- custom layout managers
- a date chooser panel
- serialization utilities

JCommon is licensed under the terms of the GNU Lesser General Public Licence (LGPL) version 2.1 or later [7].

**JFreeChart:** JFreeChart is a Java chart library for developers to display professional quality charts in their applications. JFreeChart's extensive feature set includes: a consistent and well-documented API, supporting a wide range of chart types; a flexible design that is easy to extend, and targets both server-side and client-side applications; support for many output types, including Swing and JavaFX components, image files (including PNG and JPEG), and vector graphics file formats (including PDF, EPS and SVG); JFreeChart is open source or, more specifically, free software. It is distributed under the terms of the GNU Lesser General Public Licence (LGPL), which permits use in proprietary applications [8].
Chapter 2

APPLICATION ARCHITECTURE AND DESIGN

2.1 Application Architecture

The application is designed to be user friendly, easy-to-use and compact. It is a simple application where user provides data and the application generates graph based on the user data. The application mainly has three layers as in Figure 1:

1. Presentation Layer
2. Application Layer
3. Data Layer

Figure 1. Architectural Diagram [9]
2.1.1 Presentation Layer
Presentation Layer is used to send data to the application layer in an appropriate, accurate and well defined manner. Mostly the data sent to the application layer includes user requested data and the web content. The data can be communicated in different formats from multiple sources, and the presentation layer is responsible for suitable operations like data encryption/decryption, data compression, converting the strings or handling the graphics. In the current application, the main responsibilities of presentation layer are to allow user to select specific type of date selection, enter valid date and weather parameter. It verifies the user data and sends the data to the application layer if it is valid.

2.1.2 Application Layer
Application Layer is where the actual communication is being initiated. In the current application the user provided data is verified and received from application layer. The application here is responsible to get the user requested data from the data layer and perform necessary operations. The application layer here retrieves timestamp and weather data within the user provided range from SMUD Data. The OLAP operations are performed on data which mainly helps the user to analyze data in different dimensions. The data uses JfreeChart API to generate line graph and the graph is then converted to a .PNG file and sent to the presentation layer.
2.1.3 Data Layer

The data layer provides simplified access to the data stored in persistent storage, such as database or csv file [10]. In the current application the data is stored in the form of .csv file where each row represents one record. Each record has a unique record id, time stamp and sixteen different weather parameters. Figure 2 represents the schema for data layer.

![Data Layer Schema](image)

Figure 2. Data Layer Schema

The data layer sends the data to the application layer based on the query sent by the application layer.
In the current application, the application requests for timestamp and weather parameter.

Figure 3 provides a sample query.

```
SELECT TimeStamp, WeatherParameter FROM SMUD_Data
WHERE From_date >= StartDate
AND To_date <= EndDate
```

*Figure 3. Sample Query*
Online Analytical Processing (OLAP) is used in many Business Intelligence applications. It is a powerful technology which analyses the data in multiple dimensions. It is a commonly used analytical method used for forecasting, budgeting, reporting, Knowledge Discovery and many more. It provides the user with fast and consistent access to information. Mining Weather Data application uses OLAP techniques in the retrieving the data from persistent data storage. The application uses ROLAP technique to perform various OLAP operations such as slice, dice, roll up and drill down.

3.1 OLAP Types

There are three types of OLAP techniques,

- Multidimensional OLAP (MOLAP)
- Relational OLAP (ROLAP)
- Hybrid OLAP (HOLAP)

Our application is using ROLAP technique to extract data from the persistent database. Before detailing the implementation of ROLAP in the project, it might be helpful to provide some information on all the OLAP techniques.
3.1.1 Multidimensional OLAP (MOLAP)

MOLAP is a commonly used OLAP technique. This technique is used to analyze the data which is stored in Multi-dimensional databases [11]. In general the OLAP application itself analyzes the data in multi dimensions if the data is stored in traditional databases (relational databases). MOLAP can process the data that is already stored in a multidimensional database in which the data combinations are reflected by accessing the data directly.

3.1.2 Relational OLAP (ROLAP)

As the name itself explains, ROLAP is used to access data stored in the relational or extended relational databases. ROLAP accesses the data and generates SQL queries to calculate information appropriate level when an end user requests it [12]. The ROLAP mainly relies on manipulating the data stored in databases using common OLAP operations called slicing and dicing. In SQL the slicing and dicing operations can be performed using a “WHERE” clause. The ROLAP itself has no limitation to the data; the data limit always relies on the size of the relational databases. It can leverage the functionalities that are inherited in the traditional relational databases as it comes with a host of functionalities.
The ROLAP uses one or more star schemas that are stored in databases. ROLAP is a mode of operation which uses the star schema directly as a source for the application without creating any cube structure. ROLAP is chosen as OLAP technique for the Mining Weather Data application. The current application accesses the date mart schema to access the data from database and perform operations.

Data processing can be done within the database system or server side or client side. In traditional databases, the data processing is done in a two-tiered architecture where the user submits SQL query to the database and receives data directly from the database. Figure 4 explains how the query process occurs in a two tiered architecture:

Figure 4: Two tier architecture for data processing [13]
ROLAP uses a three tier architecture where the user submits data processing request for multidimensional analysis. The architecture adds a middle layer known as ROLAP server; it converts the user request to SQL query and submits to database. The database sends the result set to a ROLAP server, where it again converts the result data from SQL result set to multidimensional format before it is returned to the user.

Figure 5 represents the three tier architecture for data processing using ROLAP server:

![Three tier architecture for data processing](image)

*Figure 5: Three tier architecture for data processing [13]*

In a relational database, few queries are created and stored in advance which helps in reducing the overhead time of running a lengthy query. If not, the query is created as user requested and run using the three tier architecture.
3.1.3 Hybrid OLAP (HOLAP)

HOLAP is a combination of both MOLAP and ROLAP [14]. HOLAP can store data in both relational databases as well as multi-dimensional databases. It uses one of the two databases based on what is best suitable for the given user request. For example, in a data heavy process, HOLAP uses relational databases as it stores data in an efficient manner, whereas it uses multi-dimensional databases while performing a speculative processing.

3.2 OLAP Operations

OLAP provides a user-friendly environment for interactive data analysis [15]. There are several number of operations that are used for data analyzing. Some of the commonly used operations are:

- Roll Up
- Drill down
- Slice
- Dice

This Mining Weather Data application uses these operations to retrieve data from persistent databases. This section details description for each OLAP operation with an example.
### 3.2.1 Roll Up Operation

Roll Up operation is mainly used for viewing the data in increased level. In Figure 6, the OLAP cube has three dimensions (1) Weather Parameter (2) Temperature and (3) Time. When the RollUp operation is performed on the cube, the content of the cube is increased by one level. The dates are now rolled up to months.

*Figure 6: Roll Up Operation [16]*
3.2.2 Drill Down Operation

Drill down operation is mainly used for viewing the data in a decreased level. In Figure 7, when the drill down operation is performed on the cube, the content of the cube is decreased by one level. The months are now drilled down to days.

Figure 7: Drill Down Operation [16]
3.2.3 Slice Operation

Slice operation selects one dimension and creates a new cube. In SQL the slice can be performed using SELECT operation with a where clause. For example, in the weather cube if the user wants to create a new cube which only has a dimension for date 01-01-2012, the SQL query can be written as follows:

```
Select Temperature, TimeStamp from WeatherData where TimeStamp == “01-01-2012”
```

Figure 8 provides a data cube for the provided example. After slice operation, the data cube only has one dimension for date “01-01-2012”.

Figure 8: Slice Operation [16]
3.2.4 Dice Operation

Dice operation is similar to Slice operation. In dice operation, the new sub cube can be created with two or more dimensions. Typically in SQL SELECT statement, a WHERE clause with OR condition can be substituted for a dice operation. For example, in the weather cube if the user wants to create a new cube which only has a dimension for date “01-01-2012” OR “01-02-2012”, the SQL query can be written as follows:

```
Select Temperature, TimeStamp from WeatherData
where TimeStamp == “01-01-2012” OR “01-02-2012”
```

Figure 9 illustrates the given example:

*Figure 9: Dice Operation [16]*
Chapter 4

IMPLEMENTATION

The Mining Weather Data is a web application which is designed using HTML, JavaScript, JFreeChart and MSSQL technologies. The application is developed using NetBeans IDE which supports writing code in HTML, JAVA and JavaScript for the current project. The database for the project is designed and maintained using MSSQL, the data is stored using star schema.

4.1 Data Mart Implementation

Star Schema is used to implement the data mart for the Mining Weather Data application. The star schema is the simplest data mart schema which consists of one or more fact tables referencing to any number of dimension tables [17]. The application has one fact table and three dimension tables. Table 2 provides a list of tables used for the project and also tells the type of the table.

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Type of the table</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbo.SG_DEVICELOG_SI</td>
<td>Fact table</td>
</tr>
<tr>
<td>dbo.SG_LOGRECORDINFO</td>
<td>Dimension table</td>
</tr>
<tr>
<td>dbo.SG_TIMEFORMATS</td>
<td>Dimension table</td>
</tr>
<tr>
<td>dbo.SG_WEATHERINFO</td>
<td>Dimension table</td>
</tr>
</tbody>
</table>

*Table 2. Tables Used*

The project uses four tables where dbo.SG_DEVICELOG_SI is a fact table with dimensions dbo.SG_LOGRECORDINFO, dbo.SG_TIMEFORMATS and Dbo.SG_WEATHERINFO.
The fact table typically has two types of columns: 1. Primary key columns that contain facts 2. Foreign keys to dimension tables. Figure 10 describes the data mart design for the Mining Weather Data application:

![Star Schema for Mining Weather Data](image)

**Figure 10: Star Schema for Mining Weather Data**

Star schema allows user to detail the level of facts that have been aggregated. In the above figure it is shown that data is sliced across all the dimension tables, and yet it is possible for the data to be aggregated across dimension tables. Dbo.SG_DEVICELOG_SI is a fact table that can get the aggregated data across all the dimension tables as following:

- Weather information for a RecordId
- DEVICE_ID for a RecordId
- TIMEDATE for a TimeStamp
4.1.1 Sample Queries Used

The application uses several queries to retrieve data from the database. Some of the queries used are connect, create, select etc.

**Connect:** For accessing data from the database, the application needs to have a successful connect statement. On a successful connection establishment the user will be able to send request and receive response from the database. Figure 11 provides a sample connect statement to the MSSQL Server.

```c
var ConProv = new ActiveXObject("ADODB.Connection");
var ConnString = "Driver={SQLServer};Server=serverName;Database=dbname; UID=thouta;PWD=pwd";
ConProv.Open(ConnString);
```

*Figure 11: Connect Statement*

**Create:** The structure of the data needs to be created before using it in the application. As mentioned above, the data mart of the application has four tables with different columns and data. Figure 12 provides a create statement used for creating Dbo.SG_DEVICELOG_SI table.

```sql
CREATE table Dbo.SG_DEVICELOG_SI
(
    TimeStamp DATETIME NOT NULL,
    RecordId INT NOT NULL
)
GO
```

*Figure 12: Create Statement*
**Select:** Select statement returns the data from one or more tables that satisfies the provided condition. The application uses select statement to get timestamp and weather data requested by the user. For example, if the user chooses to get records for Air temperature for the dates between Apr 17\textsuperscript{th} 2012 to Apr 20\textsuperscript{th} 2012, the sample query will be as Figure 13,

```sql
SELECT a.TimeStamp, b.Air_Temperature
FROM Dbo.SG_DEVICELOG_SI a, Dbo.SG_WEATHERINFO
WHERE From_date >= 2012-04-17
    AND To_date <= 2012-04-20
    AND a.RecordId == b.RecordId
```

*Figure 13: Select Statement*

**4.2 User Interface Implementation**

The User Interface for the Mining Weather Data Application is designed using HTML, JavaScript and CSS. The application has a simple interface where the user can choose Date Selection or Range Selection to choose the type of date. If the user chooses Date Selection, the application allows the user to enter only one date for which data needs to be retrieved. If the user chooses Range Selection, the application allows user to enter start date and end date for which the data needs to be retrieved. The application also allows the user to choose weather parameter where the data is chosen from one of the sixteen different weather parameters.
The following are the screens of the application:

**Screen 1: Home Screen**

![Figure 14: Home Screen](image)

Figure 14 provides a screen to let the user enter data and generate graphs for a given date and weather parameter.
Figure 15: Graph for Range Selection

Figure 15 provides the graph that is generated for the user provided data. Here the Y-axis represents the Press_Average and X-axis represents the date range. The time range for the graph here varies according to the user provided range. The range can vary from days to months to years.
Figure 16: Graph for Date Selection

Figure 16 provides the graph that is generated for the user provided data. Here the Y-axis represents the Press_Average and X-axis represents the time range. The time range for the graph here varies in minutes to hours.
4.3 Implementation of OLAP in Mining Weather Data Application

The Mining Weather Data application is using ROLAP technique to analyze the weather data from the California Smart Grid Center. The main reason for using ROLAP is because a relational database is used as back-end of the application. The user enters the data, the html transfers this data to the application layer where it performs the four types of OLAP operations and generates a graph using JFreeChart API. The process flow for OLAP technique is as Figure 17:

![Figure 17: Process flow for OLAP technique](image)

The process flow of ROLAP technique used in the application mainly performs eight steps:

**Step1:** In this step the user enters data for generating the graph. Typically the data includes start date, end date and weather parameter. There are sixteen different weather parameters available in the Mining Weather Data application.
Step2: The HTML verifies the validity of the data entered by user and sends the formatted data to the ROLAP server. For example, if the user enters invalid date in the application, the HTML wouldn’t allow the process to go further until the user enters valid data. Once user enters valid date, it will send the start date, end date and weather parameter in the form of valid query to the ROLAP server.

Step3: The ROLAP operation includes slice or dice operation, converts the query to multidimensional query and sends it to the Database.

Step4: The database generates the query and sends all the data that are applicable to a provided query. For example, if the start date given is 01-01-2012 and end date is 01-10-2012, the database sends the timestamps and weather data that lies within the given date range.

Step5: The ROLAP server performs necessary drill down or Roll up operations and analyzes the data in multiple dimensions. In the current application if the user enters the date range that is varying by a day, the application is drilled down to hours. In the same manner if the user enters date range that is varying by years, the application performs rollup operation to get data in years.

Step6: The ROLAP server sends the resulting multidimensional data to the JFreeChart API for generating the graphs.
**Step 7:** The application uses JFreeChart API and generates the graphs. The application is designed to generate Time Vs Weather parameter graph where the weather parameter can be one of the sixteen parameters such as air temperature, Press average, wind speed etc.

**Step 8:** The JFreeChart API then sends the .PNG file to the HTML document for displaying it to the end-user.

4.4 JFreeCharts Implementation

JFreeChart is a Java chart library which is used for displaying charts in the current application. It supports variety of charts such as pie chart, bar chart, line chart, timeseries chart, histograms and Gantt charts. The application provides line graphs which typically has the TimeStamp on X-axis and Weather parameter on Y-axis. The application provides two types of charts:

1. Weather data Vs Date Range
2. Weather data Vs Time Range

The application provides the user to either generate a graph for a specific day or for specific range of days. If the user chooses a specific day, the graph varies with respect to time of a certain day, and if the user chooses specific date range the graph varies with the days specified in the given range.
The following code provides user to create a sample chart using JFreeAPI

```java
JFreeChart chart = ChartFactory.createTimeSeriesChart(
    "Weather Chart", "time Range("+sdate+" - "+edate +"),hm.get(cname),
    dataset, false, false, false);
    XYPlot plot = (XYPlot) chart.getPlot();
    DateAxis domain = (DateAxis) plot.getDomainAxis();
    domain.setDateFormatOverride(DateFormat.getDateInstance());
```

Once the application creates a graph, it is written to the HTML page as JPEG file. The statement used for writing the chart as JPEG is as follows:

```java
ChartUtilities.writeChartAsJPEG(response.getOutputStream(),chart,1000,500);
```

When a user enters valid data and generates a graph, the application sends the data to the client program and performs necessary operations using Java library and JFreeChart library. The output of the client program converts the chart to JPEG file and sends to the application frame. Figure 18 provides an Application flow for JFreeChart.

---

**Figure 18: Application flow for JfreeChart**
The application frame uses HTML functionalities to display the graph. The current application uses a form tag to display this graph. The following HTML statement performs an action to display “chartshow.jsp” when the user submits valid data and generates a graph.

```html
<form name="queryform" method="post" action="chartshow.jsp" onsubmit="return validate();">
</form>
```

Some of the validity checks performed by the Mining Weather Data application are:

- User entered valid date, date field cannot be a string or an empty field
- Start date is lesser than the End date
- The date must be in the right format, the current application allows user to enter date in yyyy-mm-dd format
- A valid start date or end date is entered, i.e. the application throws an error when the user entered date is not present in the database
- User selected appropriate radio buttons

The application provides a fully functional graph, which dynamically sets the range for X-axis and Y-axis based on the input provided. The range is set based on the input provided to the graph. The Jfreechart limits the number of nodes on X-axis and Y-axis in order to preserve the visualization of the chart.
Figure 19 represents the sample line graph generated from the application. The following graph has time range field on X-axis. The data for the time range is chosen between Apr 7, 2012 to Apr 10, 2012. On the Y-axis the range is chosen between 7-26. The nodes are set according to the lowest and highest values of the data.

Figure 19: Weather Chart
4.5 Application Flow

The Mining weather data has a simple application flow where the user enters data and the system generates a graph by performing ROLAP operations in the background.

4.5.1 Application flow for User

In Mining Weather Data application, the user is responsible for providing valid data to the server. For example, the user has to enter a start date which is lesser than the end date.

Figure 20 represents the application for user.

![Figure 20: User Application flow](image-url)
4.5.2 Application flow for System

The system is responsible for validating the user data, convert the user data to ROLAP queries, perform OLAP operations, generate the graph, convert the graph to .jpeg file and send the output to user. Figure 21 describes the flow of system operations.

![System Application Flow Diagram](image)

**Figure 21: System Application Flow**
4.5.3 Application flow for Range Selection

In order to generate graphs for a specified time range, the user needs to choose a Range Selection which lets the user to enter start date and end date to generate a graph. Typically the graphs generated by a range selection have a timestamp on X-axis which varies with different dates over a calendar year. Figure 22 provides the application flow for Range Selection.

![Diagram](attachment:image.png)

Figure 22: Range Selection Application flow
4.5.4 Application flow for Date Selection

The application flow for Date Selection is similar to the Range Selection, except here the user specifies only one date for which he needs a graph to be generated. Typically the graph generated for Date selection will have TimeStamp on X-axis which varies in hours. Figure 23 provides the application flow for Date Selection.

![Diagram of Date Selection Application flow]

*Figure 23: Date Selection Application flow*
Chapter 5

USER GUIDE

The Mining Weather Data application is a web application developed mainly using HTML and Javascript. In order to run the application on local system, the system needs to have the following softwares installed.

- Java Development Kit 7 or higher
- Apache Tomcat 7 or higher

The following steps mainly describe the process steps for running the Mining Weather Data application on a local system:

**Step 1: Extract and move the application .zip folder to Apache Tomcat folder**

The application comes with a .zip folder which includes all the source code files. Before running the application, this folder needs to be moved to “webapps” folder under the Apache Tomcat folder. Figure 24 provides a sample folder location.

![Folder Location](image)

*Figure 24: Folder Location*
Step 2: Run the Tomcat Server

From command prompt, locate the bin folder where the Tomcat is installed and then run the startup.bat script to start the Tomcat server. Figure 25 provides a screen for a successful Tomcat server startup.

![Image of command prompt with Tomcat startup script output]

Figure 25: Start Tomcat Server

Step 3: Run the application using localhost server in web browser

After the successful connection of Tomcat server, the application can be run on any web browser by providing “http://localhost:8080/WeatherChart” in the address bar.
Figure 26 provides a sample procedure to run the localhost on any system.

![Sample localhost address](image)

**Figure 26: Sample localhost address**

On a successful connection, the web page will be loaded as Figure 27:

![Successful Web page](image)

**Figure 27: Successful Web page**
Step 4: Enter User Data

The user can either choose to generate graph for a given date range or for a specific date.

If the user chooses Range Selection, a valid start date and end data must be provided as Figure 28:

![Figure 28: Range Selection Screen](image)

If the user chooses Date Selection, a valid date needs to be provided as Figure 29:

![Figure 29: Date Selection Screen](image)

Once after the user enters valid date or date range, a weather parameter for which the graph needs to be generated has to be chosen. Figure 30 provides a dropdown menu for choosing weather parameter.

![Figure 30: Weather Parameter Selection Screen](image)
The graph can be generated by clicking the Generate Chart button after providing a valid user data. Figure 31 provides a screen for Generate Chart button.

Figure 31: Generate Graph

The Generate Graph button validates the user data and generates a graph as in Figure 32:

Figure 32: Sample Graph
Chapter 6

CONCLUSION

The Mining Weather data application has been successfully completed to work as intended. The application is designed for the California SmartGrid Center in order to visualize the weather data in the form of graphs. Though the application accomplishes the idea of generating graphs, the bigger scope of the California SmartGrid Center research is to optimize the power consumption generated from non-renewable resources and encourage the usage of renewable resources such as energy generated from wind, temperature and many more.

In order to serve the purpose of using renewable resources, the Mining Weather Data application helps in providing day-to-day patterns of the past weather data. These day-to-day patterns are generated in the form of graphs which can be used by California SmartGrid Center for analyzing the weather data.

In future, the application can be extended to automatically analyze the patterns and notify the electricity delivery systems whether to use a non-renewable resource or a renewable resource. Certain computer models can be generated by the system to analyze the trend of weather and send commands to the electricity delivery systems accordingly.
Also the data used for generating the graph can use an automated ETL process where the data from sensors can be saved to the database automatically. The other aspects of the application which can be extended in the future may include:

- User authentication to the website
- Generate multiple types of graphs like pie chart, gantt chart and display these charts on one screen
- Integration of Mining Weather Data application to California SmartGrid Center
- Enhance the robustness of the application

These are some of the useful ideas which can be included in the future to have it better serve for SmartGrid Center Research that helps in optimizing the usage of non-renewable resources.
Source code for Mining Weather Data application has several programs written in Javascript. Following list shows some of the important programs used in creating the application.

1. Index.jsp
2. GetData.jsp
3. ChartShow.jsp

A.1 Index.jsp

The functionalities of Index.jsp mainly includes:

- Creating User Interface
- Validating user data
- Hashing data from database to JfreeChart
- Provide input nodes to ChartShow

Source Code for Index.jsp

```jsp
<%@ page import="java.util.*" %>

Map<String,String> hm=new HashMap<String,String>();
hm.put("6","Global Average");
hm.put("7","Direct Average");
```
hm.put("8","Diffuse Average");
hm.put("9","Air Temp Average");
hm.put("10","RH Average");
hm.put("11","Press Average");
hm.put("12","WS Ms -I");
hm.put("13","WS Ms II");
hm.put("14","WS Ms Max");
hm.put("15","Rain MM");
hm.put("16","DNI Bird Average");
hm.put("17","Gen Deg Average");
hm.put("18","Global Raw Average");
hm.put("19","Direct Raw Average");
hm.put("20","Diffuse Raw Average");
hm.put("21","PSP Average");

<html>
<head>
<script type="text/javascript" src="GetData.jsp">
function validate()
{
    var date1=document.queryform.sdate.value;

</html>
var date2=document.queryform.edate.value
var b1=validateDateFormat(date1);
if(b1==false)
{
    alert("invalid date format");
    return false;
}

var b2=validateDateFormat(date2);
if(b2==false)
{
    alert("invalid date format");
    return false;
}

var start_date = getNewDate(date1);
var end_date = getNewDate(date2);
one_day_in_milliseconds = 1000*60*60*24;
date_diff = Math.floor((end_date.getTime() - start_date.getTime())/one_day_in_milliseconds);
if(date_diff<0)
{
alert("end date must be later to start date");
function getNewDate(dateString){
    var date = dateString ? new Date(dateString) : new Date(),
    timezoneOffset = date.getTimezoneOffset();
    return new Date( date.getTime() + timezoneOffset*60*1000*(dateString?1:0) );
}

function validateDateFormat(dateVal){
    var dateVal = dateVal;
    if (dateVal == null)
        return false;
    var validatePattern = /^((\d{4})((\-/|\-)(\d{1,2})((\-/|\-)(\d{1,2})\$)/;
    dateValues = dateVal.match(validatePattern);
    if (dateValues == null)
        return false;
    var dtYear = dateValues[1];
    dtMonth = dateValues[3];
    dtDay= dateValues[5];
    if (dtMonth < 1 || dtMonth > 12)
        return false;
    return true;
}
return false;
else if (dtDay < 1 || dtDay > 31)
    return false;
else if ((dtMonth==4 || dtMonth==6 || dtMonth==9 || dtMonth==11) && dtDay ==31)
    return false;
else if (dtMonth == 2)
{
    var isleap = (dtYear % 4 == 0 && (dtYear % 100 != 0 || dtYear % 400 == 0));
    if (dtDay> 29 || (dtDay ==29 && !isleap))
        return false;
    }
    return true;
}
</script>
</head>
<body bgcolor="#9911aa" align="center">
<form name="queryform" method="post" action="chartshow.jsp" onsubmit="return validate();">
    <span style="font-size:34">Weather Report</span>
    Date format is yyyy-mm-dd
</form>

Start Date : &nbsp; &nbsp;<input type="text" name="sdate" /><br><br>
End Date : &nbsp; &nbsp;<input type="text" name="edate" /><br><br>
Parameter : &nbsp; &nbsp;<select name="cname">

<% for(String ks:hm.keySet()) %>
{

<% }

<option value="<%=ks%>" value="<%=hm.get(ks)%>">

<% }

</select><br><br>

<Input type="submit" value="Generate Chart">

</form>

<body>

</html>
A.2 GetData.jsp

The functionalities of GetData.jsp mainly includes:

- Establishing connection to the SQL Server
- Run SQL queries

**Source Code for GetData.jsp**

```jsp
<%@ page import="java.util.*" %>

<html>
<head>
<script type="text/javascript">
function getdata()
    var objConnection = new ActiveXObject("adodb.connection");
    var strConn = "driver={sql server}; server=.; SQLExpress; database=smartgrid; uid=sa;password=123456";
    objConnection.Open(strConn);
    var rs = new ActiveXObject("ADODB.Recordset");
    var strQuery = "SELECT * FROM dbo.SG_DEVICELOG_SI a, Dbo.SG_WEATHERINFO b where a.RecordId == b.RecordId";
    rs.Open(strQuery, objConnection);
</script>
<body>
</html>
```
A.3 ChartShow.jsp

The ChartShow.jsp is mainly used for generating the graph.

Source Code for ChartShow.jsp

```jsp
<%@ page contentType="image/jpg" %>
<%@ page import="org.jfree.chart.*" %>
<%@ page import="org.jfree.chart.plot.*" %>
<%@ page import="org.jfree.data.xy.*" %>
<%@ page import="org.jfree.chart.axis.*" %>
<%@ page import="org.jfree.chart.plot.*" %>
<%@ page import="org.jfree.chart.*" %>
<%@ page import="org.jfree.data.time.*" %>
<%@ page import="java.util.*" %>
<%@ page import="java.text.*" %>
<%@ page import="java.io.*" %>

Map<String,String> hm=new HashMap<String,String>();
hm.put("6","Global Average");
hm.put("7","Direct Average");
hm.put("8","Diffuse Average");
hm.put("9","Air Temp Average");
hm.put("10","RH Average");
```
hm.put("11","Press Average");
hm.put("12","WS Ms -I");
hm.put("13","WS Ms II");
hm.put("14","WS Ms Max");
hm.put("15","Rain MM");
hm.put("16","DNI Bird Average");
hm.put("17","Gen Deg Average");
hm.put("18","Global Raw Average");
hm.put("19","Direct Raw Average");
hm.put("20","Diffuse Raw Average");
hm.put("21","PSP  Average");

(String sdate=request.getParameter("sdate");
String edate=request.getParameter("edate");
String cname=request.getParameter("cname");

final TimeSeries series = new TimeSeries("Series1", Millisecond.class);

try
{
    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
    SimpleDateFormat fdf = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss");
}
File f=new File(application.getRealPath("/")+"/data/smud-wt_weather_data.csv");
System.out.println(f.getAbsolutePath());
FileInputStream fis=new FileInputStream(f);
BufferedReader br=new BufferedReader(new InputStreamReader(fis));
String line=null;
br.readLine();
br.readLine();
int i=0;
while((line=br.readLine())!=null)
{
    System.out.println(line);
    String tokens[]=line.split(",");
    String odate=tokens[2];
    //System.out.println("only date:")+pdate);
    Date ddate=dfdf.parse(pdate);
    Date isdate=sdf.parse(sdate);
    Date iedate=sdf.parse(edate);
    System.out.println("final dates"+ddate+" "+isdate+" "+iedate);
if((iedate.compareTo(ddate) >=0) && (ddate.compareTo(isdate) >=0))
{
    System.out.println(ddate + " +ddate.getTime() + " +tokens[Integer.parseInt(cname) - 1] + " -
---------- ");

    // series.add(ddate.getTime(), Double.parseDouble(tokens[8]));

    //series.add(new Millisecond(ddate), Double.parseDouble(tokens[8]));

    series.addOrUpdate(new Millisecond(ddate),

    Double.parseDouble(tokens[Integer.parseInt(cname) - 1]));

    i++;
}

else
{
    if(i>0)
    break;
}

}

catch(Exception e)
{
    e.printStackTrace();
}
final XYDataset dataset=( XYDataset )new TimeSeriesCollection(series);

JFreeChart chart = ChartFactory.createTimeSeriesChart(
    "Weather Chart", "time Range("+sdate+" - "+edate +"),hm.get(cname),
    dataset, false, false, false);

    XYPlot plot = (XYPlot) chart.getPlot();

    DateAxis domain = (DateAxis) plot.getDomainAxis();
    domain.setDateFormatOverride(Date

    ChartUtilities.writeChartAsJPEG(response.getOutputStream(),chart,1000,500);

    /* JFreeChart timechart = ChartFactory.createTimeSeriesChart(
        "Weather Report",
        "time Range("+sdate+" - "+edate +"),cname,
        dataset,
        false,
        false,
        false);
    ChartUtilities.writeChartAsJPEG(response.getOutputStream(),timechart,1000,500);*/

    //CREATE DATASET.

    //CREATE CHART.

    boolean legend = true;
boolean tooltips = true;
boolean urls = false;

JFreeChart chart = ChartFactory.createXYLineChart("Weather Report Chart","time Range("+sdate+" - "+edate + ")",cname,xyDataset,
PlotOrientation.VERTICAL,legend,tooltips,urls);

//CREATE OUTPUT STREAM.
ChartUtilities.writeChartAsJPEG(response.getOutputStream(),chart,1000,500);

response.getOutputStream().flush();

%>
BIBLIOGRAPHY


