PROPOSED DATA WAREHOUSING SOLUTION FOR
INTERNET MAPPING-RELATED INFORMATION
USING A TEMPORAL DATABASE

An Integrative Project
Presented to the
Department of Information Systems and Computer Science
Ateneo de Manila University

In Partial Fulfilment
of the Requirements for the Degree of
BACHELOR OF SCIENCE IN COMPUTER SCIENCE

By
Ian B. Clarito
Richard D. Viñas
February 2004
PROPOSED DATA WAREHOUSING SOLUTION FOR
INTERNET MAPPING-RELATED INFORMATION
USING A TEMPORAL DATABASE

Ian B. Clarito
Richard D. Viñas
Department of Information Systems and Computer Science
School of Science and Engineering
Ateneo de Manila University
13 February 2004

Advisers:
Gino L.V. Ledesma
William Emmanuel S. Yu
Pierre U. Tagle, Ph.D.

The growth of the Internet has brought about the need for network interconnectivity analysis. Researchers are interested to know how the Philippine Internet evolves. Internet Service Providers (ISPs) are interested in this data in order to provide better quality of service to end-users. However, this data will need to be organized efficiently so that all kinds of analysis can be done in the least amount of time.

This project developed a data warehouse using a relational database structure that will be used for organizing the interconnectivity data of Autonomous Systems (ASes) in the Philippines. This database can then be used for any type of analysis in the future.

Keywords: AS Availability and Connectivity, Data Warehousing, Internet mapping, Philippine Internet, Temporal Database
Chapter 1

Introduction

1.1 Background and Motivation

Proper data storage is a primary concern of all organizations. With the growth of internetworks around the world, there has been a demand for a facility to store data coming from the Internet. Given that an autonomous system (AS) is an administrative entity that regulates the Internet in a particular area which includes big schools and ISPs (i.e. Ateneo de Manila University, Mozcom, etc.), researchers and ISPs have been interested in getting internet topology data, which would provide information about how ASes interconnect to form larger networks that span the world. Analyzing the AS interconnectivity and availability information from this data would have significant impact in providing for better quality of service to the end-user. The information would be crucial for ISPs in decisions such as the purchasing of better network equipment for ISP interconnections or on new interconnections for load balancing.

The Internet has evolved from its beginnings as a small network used for military and academic purposes to its current state as an on-line global community, connecting peoples and data. The ISPs interconnect to form larger networks and this not only provides faster access to information but also connection reliability because multiple possible paths between ISPs enable data to be sent through various routes in the event of a connection going down.

Currently, there are only a handful of projects that are dedicated to storing internet topology data. These projects use either a file repository system or a private object-oriented database that are hard to manipulate because they are structured only for a specific purpose. In the Philippines, researchers have compiled an autonomous system number (ASN) list detailing information about the major Internet nodes in the country, as well as an IP Block Listing that each service provider
manages. [11] Other than this, there have been a few data analysis projects concerning AS connectivity and availability information in the Philippines and these also use project-specific data.

1.2 Statement of the Problem

The Philippine Internet needs to be mapped and the data is already available in the form of BGP (Border Gateway Protocol) data. Recent projects (see Related Works) have already provided the data regarding the inter-connections of different Autonomous Systems. The data is updated every five (5) minutes. However, there is a need for an efficient way of storing these data so that queries about availability and connectivity of ASes and graphing the Philippine Internet would be efficient. The present projects only produce the latest map of the Philippine Internet. Map analysis is limited through visual examination of the map. The challenge is to make an efficient way of storing the Philippine routes and its history, which will make any kind of analysis easier.

1.3 Objectives

The major aim of this project is to provide an effective way of storing and retrieving the BGP data. Specifically, it aims to (1) design and implement a data warehouse using a relational database that will store the BGP routes, that are, the interconnections between Philippine ASes. (2) The project also aims to produce a user-friendly web interface that will show the graphs generated from the connectivity data and answer queries about availability of Autonomous Systems.

1.4 Scope and Limitations

The system is designed for large route data. However, the project is only focused on mapping Philippine Internet data. Connections with ASes abroad are not the primary concern of the project. The main part is the database module, with real-time visualization a secondary aim of the
project.

The project is only tested for one (1) month’s worth data. We are also limited to using GraphViz as the Visualization tool since most of the visualization tools are proprietary. The system will only be tested in a computer set-up with hardware limitations. If the system works well in this environment, it is expected to perform better in other set-ups.

The routes are dependent on the efficiency and reliability of the collection of the peers. In this case, the data is dependent on the reliability the dump collector server in PREGINET.
Chapter 2

Review of Related Works

2.1 Data Warehouse

The term “data warehouse” was first coined by Bill Inmon, who defined it as “...a subject-oriented, integrated, time-variant and non-volatile collection of data in support of management's decision making process.” [5] Thus, it is a continuous process of collecting current and historical data about specific subjects and merging them into one coherent whole to be used for analysis. A characteristic of data warehouses is that due to the large amount of data, only certain periods in history are kept in the warehouse at a particular time. For instance, a year's worth of data is loaded into the warehouse and the oldest month is discarded once the newest month is completed. A single-subject data warehouse is called as a data mart.

The entire data warehousing system involves processes such as: source system identification wherein the appropriate data for the warehouse must be located. It might be retrieved from On-line Transaction Processing (OLTP) systems such as relational databases or from archived data available from websites; the data warehouse design and creation which deals with the design of the data warehouse database to ensure support for the types of queries the user will be asking in the future; data acquisition and it involves storing the data into the database using Extract/Transform/Load (ETL) tools. Data is retrieved from the source and undergoes the necessary transformation (i.e. converting data to a specific format) before loading into the database. All available data is loaded into the database and future data will be retrieved using the ETL tools as soon as they are available; and the business intelligence, which assumes that the data has been loaded into the database and analysis can begin. There are four types of tools used for data analysis: (1) Multi-dimensional Analysis Tools
which enables viewing of multi-dimensional databases from different perspectives; (2) Query tools which enables the use of Structured Query Language in getting result set of queries; (3) Data Mining Tools which enables the searching for data patterns; and (4) Data Visualization Tools that enables data representation via graphs and charts.[13] The figure bellow shows the data warehousing process.

![Data Warehousing Process Diagram]

**Figure 2.1 Data Warehousing Process**

### 2.2 Internet Mapping

Currently, there are a number of projects pertaining to the mapping of the Internet, specifically on visualization of the Internet. The Cooperative Association for Internet Data Analysis (CAIDA) has a tool called Skitter that is used for actively probing the Internet in order to analyze topology and performance. It has a visualization tool for network connectivity and other analysis tools. [3] However, CAIDA and its tools visualize the worldwide Internet. Since the Philippines is only a point in the global internet map, it is hard to do analysis of data using their present tools.

The University of Oregon Route Views Project also has the same project as CAIDA. It is concerned with the global routing system from the perspectives of several different backbones and locations around the Internet.[1] It does not include the Philippines in its analysis.

The Netlantis project is the former zebra.swinog.ch and mogwai.frnog.org BGP RouteViewer project. Netlantis is an open research project for real-time analysis and representation of BGP routing tables. They are involved with the collection of the global BGP data which can by
analyzed using their available tools such as the Graphical AS Path (GASP) which shows of the interconnectivity of ASNs starting from specific prefix of an ASN and the AS Monitor which shows the history of appearing and disappearing ASNs.

Their system runs on over 100 peers and the database has stored over ten million prefixes. They maintain one table per BGP peer and use a "merge" type table in order to do the queries overall the tables. Their current set-up is an Intel dual Xeon 1.8Ghz with 2Gb of RAM and UltraSCSI-160 disks. One hard disk is fully dedicated to the database management system. [15]

The Routing Information Service (RIS) of Réseaux IP Européens provides information about BGP routing. However, it provides historical information about Internet routing without being bound to the perspective of a particular autonomous system. The service collects routing information by using Remote Route Collectors at different locations around the world and integrates this information into a comprehensive view. The RIS collects routing information between Autonomous Systems (AS) and their development over time. This allows the users to see the full picture with all routes that are currently anywhere and their development over time.[14]

As for the Philippines, mapping its Internet topology is a new project. Right now, the Ateneo de Manila University and the Advanced Science and Technology Institute of the DOST are spearheading the initiative. The current system includes tools for visualizing the current state of interconnectivity among the different ISPs in the Philippines. It derives the map from the Global Internet Routing Table using the Border Gateway Protocol (BGP). BGP is an inter-Autonomous System routing protocol. The primary function of a BGP speaking system is to exchange network-reachability information with other BGP systems. The limitation of this system is that in only gives the current state of the Philippine Internet. This means that when a user wants to see the graph of the Philippine internet, it re-parses and re-graphs the BGP data so that it takes at least 15 minutes for a graph to be shown. By this time, there are already changes in the data in that span of time.
Moreover, it does not show the changes in the interconnection between Autonomous Systems, the user needs to see two graphs and compare them by looking at their difference in order to see the changes. This is very difficult since the Philippine Internet is rapidly growing and the interconnection between them is also increasing. [8]

\[ \text{2.3 Temporal Database} \]

Databases can be classified into two kinds: conventional and temporal databases. Conventional Databases stores only the current state of particular information; no past information is kept. In this kind of database, transactions evolve from one state to the next without keeping a record of changes or previous state. This is also called a snapshot database. On the other hand, temporal databases maintain historical information of the data. Records have start time and end time. In this case, database evolves in time with previous states preserved.

There are three kinds of Temporal Databases. First is the Transaction Time Database whose purpose is to record the statuses at different times in history. Each record inserted is viewed as a transaction whose ending time is unknown at insertion time. Here, time evolves discretely. When an object is inserted, its temporal field has the form \([t_{\text{ins}}, \text{now})\) indicating that the ending time is still unknown. It is important, to note that updates are allowed to be made to records of the most recent version only. No alteration of the past data is allowed. Also, deletion of a record is logical. Physical deletion of the record defeats the purpose of storing the historical information about that record. Records are theoretically deleted when they are already supplied with the ending time. [10] The graph below is an example of a Transaction Time Database. Time Stamp (time slice) queries and Interval (time-series) queries can be done in this type of database. [6]
The next kind is Valid Time Database which is used to support interval data. Here, time evolves continuously. Each record is a line segment representing its time span which means that the time end is already known. Also, deletion is physical, that is, a value can be changed at any time by one deletion of a record and insertion of another. This makes it impossible to know about the previous states of intervals. It can be said that the notion of future, present, and past is relative to a particular timestamp t. It is in the future if the record time is after t, in the present if the time span contains t, and in the past, if otherwise. Like the Time Transaction Database, Time Stamp and Interval queries are also efficient. A visualization of this database is shown in the figure below.

Figure 2.3 Valid Time Temporal Database

The last type is the Bitemporal Database. This is just the same as the Transaction Time
Database but each record is an interval. This used when the need to keep the evolution of a dynamic collection of interval records arises. A single time stamp is a Valid Time Database. A picture of a Bitemporal Database is shown below. [10]

![Bitemporal Temporal Database](image)

Figure 2.4 Bitemporal Temporal Database
Chapter 3

Methodology

3.1 Overview

The Philippine Internet Mapping Server gets routes from the ASTI-PREGINET RouteViews Server, a server that consolidates all BGP routes. These routes from different sources and protocols are put in a Routing Information Base (RIB). There are two kinds of BGP RIBs that the NetMapper outputs. One is the complete connectivity information of all Autonomous Systems in a given time period (base file), and the other is an update of changes in the connectivity of Autonomous Systems in smaller time increments within each time period (update file).

The NetMapper retrieves the BGP RIBs in binary format and converts them to parser-friendly text files. These files contain data about the Autonomous Systems and to the nodes they are connected with. After the data is converted to text files, the Philippine ASNs are then extracted from the files. Only the information about Philippine ASes is supposed to be placed into the database.

In order to create the proposed database, the project will be using a base file with the complete connectivity information and the succeeding inputs will be the update files generated after the base file. This project aims to update and store the AS connectivity data to the database which comes in 5-minute intervals. After one month, the proposed system will archive the old database and begin re-gathering connectivity data.

Another module of the project is a program that gets the complete list of ASes from the internet. This is done in order to assure that the AS list and the data about each AS is up to date. This data is also inputted in the database.
After the database has been installed, a web-enabled interface will be provided for visualization and queries such as:

a. a real time graph of the current AS connections
b. a graph that shows the changes of the AS connections from a given duration of time
c. a list of Philippine ASes and some data about them
d. availability information of ASes that outputs how frequent they have been disconnected from the internet
e. list of neighbours of a given AS

3.2 Different Approaches

There are several ways of storing data. Depending on the purpose and functionalities, some are more efficient than others while others are inherently weak. We considered four different approaches in storing the data. Here are their descriptions, advantages and disadvantages.

3.2.1. Filesystem-based Data Storage and Processing Systems

This type of system concentrates on the data processing needs of individual departments instead of the entire organization. Created applications are designed to solve particular problems. One problem with this system is the uncontrolled redundancy. Since each application uses its own private files and the possibility that each use the same data, valuable disk space is wasted. Also, there will be data inconsistencies due to the large number of files needed to be updated as well as different standards enforced on the different files. This system also brings about the problem of inflexibility as seen in the case of when new formats or standards are given, all applications need to be updated which will cost time and money. Another disadvantage is limited data sharing - users will have a hard time in sharing or accessing data with other applications. [9]
3.2.2. Non-relational/Flat Databases

A major factor to satisfy users with databases is its performance compared to file-system-based storage. Databases use efficient data structures to represent the data that is why it is faster. Databases also provide security utilities, recovery control, and other functions not inherent to ordinary file-system based storage.

A flat-file database is used to manage a small collection of information. It has only one table. An example would be a class directory database listing only the names, birthdates and addresses of people. The advantage of this database lies in its simplicity wherein the user can easily add and update data without the need for complex queries. The disadvantage is slow data retrieval when the table have a lot of columns. [4]

3.2.3. Relational Databases

The relational database model is a collection of tables. The database can be modified by insertion, deletion, or update of the rows in the tables. There can be different views of the database for different users. It is important to note that we could find rigorous approaches to relational database design. This is because it is based on a firm mathematical foundation. This is one of the primary advantages of the relational models compared to other models.

There are two ways to design a relational model. One is a fully normalized and the other is semi-normalized.

a. Fully Normalized

In normalizing a database, there are four goals: to arrange data into small and logical groupings, to minimized data redundancy, to organize data for simple updating, and to be able to access and manipulate data quickly and efficiently without compromising the integrity of the data in
storage. Usually, it is important to fully normalize databases. However, in some cases such as in simple databases, a fully normalized table would perform slower than a partially normalized table.

In the First Normal Form, data is ensured to be atomic which means that each row in the database can be uniquely identified using candidate key(s), from which the primary key is chosen. In the Second Normal Form, every non-key attribute is fully dependent on the primary key. This means that the primary key can uniquely identify each non-key attribute. In the Third Normal Form, transitive dependencies are resolved. This means that when any non-key attribute that determines the value of another non-key attribute, they are placed into a new table where the determinant attribute becomes the primary key. Relations in the third normal form are sufficient for most practical database applications. For the special cases where other anomalies arise, there is the Boyce-Codd Normal Form that solves overlapping candidate keys, the Fourth Normal Form that deals with multi-valued dependency and the Fifth Normal Form that is for join dependency issues. [12]

b. Incomplete Normalization

Normalization starts with a single relation scheme and decomposes it. But sometimes, when we compute the natural join of some relations, we discover that all rows referring to some records disappear. This may occur in practical database applications in which they may represent incomplete information. The only way to write a universal relation is to include null values in the universal relation. But because of the difficulty of managing null values, it may be better to view the relations of the decomposed design as representing the database, rather than the universal relation whose scheme was decomposed in the normalization processed. [7]

Another consequence of complete normalization is that it makes the queries that require many relations slower. It takes some time to join these tables. That is why, if redundancy in tables is acceptable, many table joins are needed, it is recommended to lessen the normalization.
3.2.4. Object-Oriented Databases

Today, due to the lower costs of memory and processors, new applications require new data models, query languages and transaction models to cope with the storage complex objects that contains other objects, behavioural data that stores different responses of objects to same commands, meta knowledge which are the general rules about the application that cannot be easily represented in traditional database systems, and long duration transactions. These are the few things that Object-Oriented Database (OO database) can provide that are not available in other databases.

This type of database has become popular because of the massive adoption of object-oriented development. OO databases are important for databases that want to store non-traditional applications with non-alphanumeric data types that need persistence such as images, multimedia, and other artefacts. OO databases, like other OO systems, uses objects that encapsulate a particular data structure and operations available for it. Thus, it requires that object identity, encapsulation, dynamic binding, and types/classes are supported. Also, data persistence must be provided. The DBMS must have the capacity of handling very large databases. It must be able to support concurrent users. Some current systems are ObjectStore, Versant, and Ontos. [2]

However, relational databases have more uniformity in structure than object-oriented databases. Not all variables can fit conveniently into some structures. Some applications include highly specialized data types that are physically large and are usually manipulated using application programs that are not part of the set of methods of the classes. This makes the OO databases less flexible. Object-Oriented Queries to be processed corresponds to a relational view of the object database. We could pose queries that involve joins of objects. But, the relation view of objects is limited to the variables in the objects. Thus, OO query language leads to serious language design complications. Thus, there are currently no clear standards emerging. [7]
Chapter 4  
Architectural Design  

4.1 Description of Components

The figure above shows the different components. The proposed project is composed of three modules. The first module is the Extract-Transform-Load Module. This module is responsible for the inputting of the data into the database. This is composed of the scripts and java programs that parse the RIB and UPDATE files and insert the data into the database.
The second module is the Database module. This is the Relational Database Management System in which the temporal data is stored. Included in this module are the SQL scripts to create the tables.

The last module is the Warehouse Interface of the system. This is a user-friendly web interface that is responsible for the data visualization tools and other matters about the project. This part includes the tools for AS availability and drawing of the Philippine Internet Map.

These modules will be discussed in detail in the Implementation Details Section.

4.2 Hardware and Software Requirements

There are currently two workstations in used in the Philippine Internet Mapping Project. One is the Route-Views Server by the PREGINET that obtains the BGP routes data from different partner ISPs of ASITL. Another is the Ateneo Rack mount Server that houses the Netmapper. The proposed Data Warehousing Solution will be installed to this workstation.

The current Hardware and Software Specification of the two computers are as follows:

The Route-Views Server is an UltraSPARC-IIIi (270Mhz) with a 4103MB HDD. The software used here is zebra/bgpd. Zebra is configured as a no-login router so that it can be accessed by the public just in case they want to query the "show ip bgp" routes.

The Ateneo rack-mount Server is an Intel P4 1x1.8GHz with 512MB DDR SDRAM, 2x40 GB HDD, and 1xBroadcom 10/100 Integrated NIC. The current software installed are: Red Hat Linux 7.3; Zebra bgpd 0.92; Multithreaded Routing Toolkit (MRT) with route_btoa, route_atob components; Perl v5.x for conversion scripts; PHP v4.x for web front end; Java 2 Standard Development Kit 1.4.x; and Cron + GNU bash for automation scripts.

The use of free software for the different components is important. ETL tools were created using Java. UNIX shell scripts were used for the automated data gathering and file manipulation
while the web interface is done using PHP. Furthermore, the Database Management System that used is PostgreSQL, which is better than mySQL in terms of table joins. GraphViz, particularly Neato program, is used for data visualization. It already has available libraries for graphing.

For preliminary results, the hardware specification above is not yet used. Instead, the project was deployed in a Pentium II 350 MHz, with 256 MB RAM. The operating system used is RedHat Linux 7.3, Kernel v. 2.4.20.

Another set of hardware was used for comparison of performance. The project was also deployed in an AMD Athlon 900 MHz, with 512 MB RAM. The operating system used is Mandrake Linux 9.2, Kernel v. 2.4.22. This new hardware improved the performance of the programs.
Chapter 5

Implementation and Results

5.1 Database Design

The proposed solution is a Relational Temporal Database. A database of this kind requires table structures that use the relational model.

We started in making the relationship between the entities in needed to be stored in the database. The figure below shows the Entity-Relationship Diagram of the database.

![Entity-Relationship Diagram]

Figure 5.1 Entity-Relationship Diagram

A Subnet belongs to only one Autonomous System at a particular time period. These AS can belong to a company in a period of time. Likewise, ASes are also connected and disconnected from other ASes.
From the diagram above, we then designed the database schema like how temporal databases are modelled. For this kind of temporal database, we saw that a Transaction-Time Database is most fit to serve as the model. Thus, we were able to create the following tables shown in Table IV.1. It is important to note that each table has its time_start and time_end. This is to show temporality of the record in each row. As explained in the related works, time_start is the time when the record was active or in the present while time_end is the time when that record is no longer active. The latter timestamp is important to show that the record is already in the past and not anymore active in the present. For example, in the AS_CONNECTIONS table below, time_end tells us when the connection (the particular row) between two ASes was dropped. This time_end field has the value equal to the time_start if the record is still active presently. Same logic applies to all tables in the database.
### AS_SUBNETS

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>as_num</td>
<td>Integer(5)</td>
<td>Nnnnn</td>
<td>AS Number</td>
</tr>
<tr>
<td>Prefix</td>
<td>Varchar</td>
<td>xxx…</td>
<td>Subnet IP Address</td>
</tr>
<tr>
<td>time_start</td>
<td>Long</td>
<td>nnn…</td>
<td>Start time when the subnet was announced to belong to the AS</td>
</tr>
<tr>
<td>time_end</td>
<td>Long</td>
<td>nnn…</td>
<td>Time the subnet was dropped</td>
</tr>
</tbody>
</table>

### AS_CONNECTIONS

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>As_num1</td>
<td>Integer(5)</td>
<td>Nnnnn</td>
<td>AS Number of the First AS</td>
</tr>
<tr>
<td>As_num2</td>
<td>Integer(5)</td>
<td>Xxx</td>
<td>AS Number of the Second AS</td>
</tr>
<tr>
<td>Time_start</td>
<td>Long</td>
<td>Nnn…</td>
<td>Time the connection between the two AS was announced</td>
</tr>
<tr>
<td>Time_end</td>
<td>Long</td>
<td>Nnn…</td>
<td>Time the the Two AS was disconnected</td>
</tr>
<tr>
<td>Status</td>
<td>Boolean</td>
<td>T/F</td>
<td>T if Two AS is connected, F otherwise</td>
</tr>
</tbody>
</table>

### AS_LIST

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>As_num</td>
<td>Integer(5)</td>
<td>Nnnnn</td>
<td>AS Number</td>
</tr>
<tr>
<td>As_name</td>
<td>Varchar</td>
<td>xxx…</td>
<td>Company name where the AS belongs</td>
</tr>
<tr>
<td>Time_stamp</td>
<td>Long</td>
<td>Nnn…</td>
<td>Time when the company had the AS</td>
</tr>
<tr>
<td>Time_end</td>
<td>Long</td>
<td>Nnn…</td>
<td>Time when the AS does not belong to the company anymore</td>
</tr>
</tbody>
</table>

Legend:
- nnn… All numeric
- aaa… All alphabetic
- xxx… Alphanumeric

Table 5.1. Tables Schema 1

However, access speeds to the database are very important. That is why the schema above was improved to have an efficient way of storing the data. The first modification of the tables above is the addition of another table called AS_SUBNET_HISTORY. Since the past connections between subnets and ASes are not anymore very important to determine the connectivity between ASes, these past connections records were transferred to another table which stores the data for the purpose of history retrieval. In this case, the access to the AS_SUBNETS table becomes faster since the voluminous past records in the table were already transferred to a static table, that it, a table that
does not anymore change aside from additional past connections. The effect of this is that the AS_SUBNETS table’s time_end field was removed, since it is already assumed that all records in the AS_SUBNETS table are active in the present.

As we can see, due to the goal of having an efficient data warehouse, the normalization of the table is not perfect. The addition of the AS_SUBNET_HISTORY is an example. The final table schema used in the database is shown in the table below.
To show the improvement of the new table schema, the table below shows the changes in
the insertion time of the same rib. They vary in the time it took to insert them. The insertion in the
new schema is faster.

<table>
<thead>
<tr>
<th></th>
<th>5 KB</th>
<th>27 KB</th>
<th>3 M</th>
<th>5 M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program 1</td>
<td>7 s</td>
<td>53 s</td>
<td>44 m 50 s</td>
<td>3 h 31 m</td>
</tr>
<tr>
<td>Program 2 (with new table)</td>
<td>8 s</td>
<td>19 s</td>
<td>24 m 15 s</td>
<td>1 h 14 m</td>
</tr>
</tbody>
</table>

Table 5.3 Differences in Insertion time Between Two Table Schema

5.2 Extract-Transform-Load Tools

To insert the data into the database, there are several ETL Tools developed. Since it is very
cumbersome and tedious for the user to manually read the dumps and insert them manually into
the database, the following files and programs were created:

Configuration Files: A Property File (named netmapper.properties) and a Class (named
NetMapperProperties) that reads and loads the property file to the memory has been made in order
to ensure fast and efficient configuration of other programs. Some properties or configuration
needed and were put in the property file are: Database URL, Username, Password, and JDBC
Driver Name; Batch Size; and Log Filename.

convertUpdates.sh: A script that calls that route_btoa program converts all available update
files in binary format. The route_btoa program is responsible for transforming them in to a parser-
friendly text file. These text files are then moved into a directory.

InsertASNLList.java: This class inserts the data about the ASes written in the PH-ASN.txt
file which can be downloaded from cng.ate neo.net. The PH-ASN.txt file is only a key-value pair
where each AS in the Philippines is paired with its name or the name of the company where it
belongs. This file is not that dynamic, that is, the names of the ASes do not belong as often as the connections. Therefore, it is sufficient to insert this data once a month or so. The class parses the text file and just inserts them into the database, particularly in the AS_LIST table, with a particular time_stamp.

**RibLine.java, RibParser.java, RibContent.java, RibProcessor.java, InsertRib.java:**
The database has to start with a base data. For this matter, it needs to know the active connections in order to see the changes in the connections that happened in the succeeding rout files. This base file that has the connections at the start of the database is the RIB file. It is basically a text file that has all the “announces” of subnets active during the time it was created, and to which AS it belongs. The file includes how ASes are connected to each other. This can be seen in the AS Path field of each line in which it is shown that there is a connection between two ASes if they are consecutive in their order in the AS path. Each Rib file has some number of lines depending on the connections it saw. The documentation of each line is available in the route_btoa documentation.

The rib file (base file) is being parsed by the RibParser class. After the important information in the rib file has been extracted, a new RibLine object is created that contains the specific information in the Rib line. This RibLine object determines to which AS-Subnet relation and the immediate neighbours of that. The AS neighbour is the AS number right beside the last AS, in which the subnet belongs. The RibContent is a container of all RibLine objects in the Rib File.

The RibProcessor collects all AS-AS pairs and put them into a TRIE data structure. Here, redundant pairs, ASes connected to ASN 0 and ASes connected to itself, are discarded. The building up and use of a TRIE data structure is more efficient than inserting all the AS pairs into memory and letting the database to handle the redundancy issues. These AS pairs determines the AS connections in the Philippine Internet. If an AS is paired to another AS, then there is a connection between them.
This parsing algorithm of just getting last two AS numbers in the AS Path versus the algorithm of parsing all of each consecutive AS has been proven reliable. Implementations of both algorithms are made and it shows that they still have the same number of AS connections.

The InsertRib class finally inserts the list of AS-Pairs to the AS_CONECTIONS table, and the AS-Subnet Composition to the AS_SUBNETS table.

The use of batch inserts has been used in order to have a more efficient way of inserting data into the database. The batch size in the Property File tells the ETL tools when to inset the data. This is used in InsertRib and InsertUpdate classes.

**UpdateLine, UpdateContent, UpdateParser, UpdateProcessor, InsertUpdate:** The changes in the connections in the ASes and the changes where the subnets belongs is available in the update files. These files are available every five minutes. Basically, an update file announces the new subnets that were not available during the previous batch of announces just like how a Rib File does. However, the difference from a Rib File is that an update file contains the “announces” and drops of subnets between the time of the last update file and the time it was created. This means that the time_stamp inside an update file varies. The UpdateParser parses this file and creates UpdateLine object per line. The UpdateContent is a container of all the UpdateLine objects in an Update file. The UpdateLine class is just the same as the RibLine Class, only that it remembers the time_stamp of that line.

The InsertUpdate inserts the information in the update files into the database. We were able to test two methods of data insertion. The first approach is to update, or insert the records applicable to the UpdateLine directly. In the first approach, every time a line is being read, a query or update statement is done into the database.

The second approach is to place into memory all the active connections of ASes in the AS_CONNECTIONS table, and all the AS-Subnet pairs from the AS_SUBNETS table. In this
case, all updates and insertions are made in the memory and not in the database. Only after all update lines have been processed that the change to the database is being made. We can see in the table below that this is faster than the first one.

<table>
<thead>
<tr>
<th></th>
<th>5 KB</th>
<th>27 KB</th>
<th>3 M</th>
<th>5 M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program 1 (DB Updates)</td>
<td>9 s</td>
<td>1 m</td>
<td>51 m 18 s</td>
<td>4 h 19 m</td>
</tr>
<tr>
<td>Program 2 (Mem Updates)</td>
<td>7 s</td>
<td>53 s</td>
<td>44 m 50 s</td>
<td>3 h 31 m</td>
</tr>
</tbody>
</table>

Table IV.4. Differences in Insertion Time of two Approaches in inserting an update file

5.3 Business Intelligence Applications

The database will be used to answer relevant business questions of the target users. The web module of the project presents some of the queries of the target users. Here are the scripts used to do the said transactions:

**createCurrentMap.sh**: This shell script calls the Neato program to create the latest picture of the Philippine Internet by using the latest data. It renames the last picture of the Philippine Internet as previousMap.png. The new picture is named currentMap.png. Every five minutes, this script is called to create the newest picture. The latest picture is shown in the main page of the web module (index.php)

**viewEntireMap.php, viewSpecificAS.php**: PHP form pages that accept ASN and two dates to be used for connectivity analysis. They will call shell scripts, using the inputs as parameters, to start the analysis and present the results in the form of pictures of the Philippine Internet as well as the availability information in text format.

**createASMap.sh, createEntireMap.sh**: These shell scripts will use the dates and ASN as inputs coming from the viewEntireMap.php and viewSpecificAS.php. The said inputs will be used
to call a Neato-valid command that will generate the image file. These scripts will be calling the
ToDotFile and DBQueryer classes to create the file format used by Neato using data retrieved from
the database.

**viewAvailabilityInformation.php**: PHP form page that presents the connectivity
information of a specific AS. It will call a shell script to begin analysis using the inputs, and present
the results in the same page.

**viewAvailabilityInformation.sh**: The shell script that will use the ASN input coming from
viewAvailabilityInformation.php to call the ToDotFile and DBQueryer classes in order to retrieve
the result set that will be read by the output page.

**viewSubnets.php**: PHP form page that presents the list of subnets that belong to a specific
AS. It will call a shell script to begin analysis using the inputs and present the results in the same
page.

**createSubnets.sh**: The shell script that will use the ASN input coming from
viewSubnets.php to call the ToDotFile and DBQueryer classes in order to retrieve the result set that
will be read by output page.

**ToDotFile.java, DBQueryer.java**: ToDotFile and DBQueryer class will use the inputs
coming from the PHP form pages in creating the SQL statement to retrieve the data from the
database. These data will then be transformed into the dot format to be used by the Neato program.
Chapter 6

Observations and Analysis

6.1 Maps Generated

There are two general kinds of map that can be generated by the project. One is the Philippine Internet Map and the other one is Connection Maps for a Specific AS.

The first type is basically a map of the whole Philippine Internet. In this map, all interconnections between ASes in the Philippines and their immediate connections to foreign ASes can be shown. Each node in the graph represents an AS. Their interconnections to other ASes are represented by the edges. The edges have been colour-coded to tell whether the disconnections between ASes have exceeded five times already. The number of disconnections is specifically shown through the weight of the graph.

Two kinds of Philippine Internet Map can be generated by the system. One is the current map where all disconnections can be seen. The graph below shows the Philippine Internet Map from the start of the data acquisition to November 6, 2003, the latest date we have inserted data.
Figure 6.1 Philippine Internet Map from October 9, 2003 to October 31, 2003

Another type of the Entire map is the mapping of the Philippine Internet during a specific time period. An example of this is shown below which is a graph of the Philippine Internet from October 13 to October 13, 2003.
A less general map can also be drawn by the system. Specific AS and their immediate neighbours can be seen in this kind of map. The same with the entire map, this can also be done in two ways. Those are: the connections of the AS from the start of data acquisition; and the connections of the AS during a time period. The map below shows the connection of AS 4775 (GLOBE-TELECOM-AS) from October 13 to October 22, 2003.
Figure 6.3 Graph Showing AS 4775's Neighbours from October 9, 2003 to October 22, 2003

6.2 Speed

The table below shows the time it takes to insert the Update files with different file sizes:

<table>
<thead>
<tr>
<th>Program 2 (Mem Updates)</th>
<th>5 KB</th>
<th>27 KB</th>
<th>3 M</th>
<th>5 M</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>7 s</td>
<td>53 s</td>
<td>44 m 50 s</td>
<td>3 h 31 m</td>
</tr>
</tbody>
</table>

Table 6.1 Insertion Speeds of the Update Files

From these data, it can be noted that the time it takes to insert the update routes is highly dependent on the size of the update files.

The same observation was derived in querying the database. The table below shows the amount of time it takes to query the whole Philippine Internet Connections using the two different sets of hardware.
<table>
<thead>
<tr>
<th>Hardware Specification</th>
<th>Pentium II 350 MHz, 256 MB RAM</th>
<th>AMD Athlon 900 MHz, 512 MB RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time it takes to query</td>
<td>4 m 30 s</td>
<td>50 s</td>
</tr>
</tbody>
</table>

Table 6.1 Query Speeds Using Two Different Hardware

The values above are based on the performance of the querying when the computer is not doing any process that require large amount of processing power. The drawing of this information by the neato program takes 10 seconds for both hardware specifications.

The use of a more powerful hardware may have improved the performance of the querying of the entire map from the database. However, according to the set standards, it is still slow. In the current architecture, every time a query of the map is needed, a program connects to the database a queries a table of many rows. This is what makes the production of the dot file too slow.

In order to solve this, a new approach was implemented. In the new approach, only one program connects to the database. Then, it builds all the data using data structures in the memory. All requests for the connectivity data come from these data in the memory and not anymore directly from the database. This was implemented using multi-threading and sockets.

If the processing is uninterrupted, the initial loading of the data from the database to the data structure takes almost four minutes. However, there is an improved performance when the production of the dot file. Apparently, it only takes one (1) second to produce the dot file, another ten (10) seconds to draw the map. Thus, using this approach, drawing the whole Philippine Internet Map only takes eleven (11) seconds.

6.3 Analysis
From October 9, 2003 6:51 p.m. to November 7, 2003, the following sets of information were derived: There are three major Internet Exchange points in the Philippines that have the most number of connections. First is AS 9299 named IPG-AS-AP. This has 16 ASes connected to it. Second is AS 4775, GLOBE-TELECOM-AS, with 15 AS connections. And third is AS 9670, the MANILA-IX, which has 12 connections.

It can also be noted that the following ASes have the highest number of disconnections in this month:

<table>
<thead>
<tr>
<th>Rank</th>
<th>AS 1</th>
<th>AS Name</th>
<th>AS 2</th>
<th>AS Name</th>
<th>Disconnections</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>23770</td>
<td>PNOC-EDC-AS-PH</td>
<td>6163</td>
<td>ERX-MOZCOM-NETWRK</td>
<td>141</td>
</tr>
<tr>
<td>2</td>
<td>18396</td>
<td>PHILCOMCORP-MND-AS-AP</td>
<td>4775</td>
<td>GLOBE-TELECOM-AS</td>
<td>58</td>
</tr>
<tr>
<td>3</td>
<td>7551</td>
<td>BANCNET-AS-PH-AP</td>
<td>4775</td>
<td>GLOBE-TELECOM-AS</td>
<td>37</td>
</tr>
</tbody>
</table>

Table 6.3 ASes with the Highest Number of Disconnections

There are some ASes that are connected only to its service provider. Furthermore, a good number of these ASes were frequently disconnected. This means that the ISP does not provide reliable service. It would have to upgrade its network equipment or else its customers will be better off in moving to another provider.

On the other hand, there are ASes which have not been disconnected throughout the entire month. This indicates the high availability of these ASes in the Philippine Internet map.
Chapter 7

Conclusion and Recommendations

The project is a data warehousing solution, a data mart in particular, in the sense that Internet topology data from the world is stored into the relational database. The data is gathered from a variety of sources and they give historical information that can be used for business queries. All data are associated with the time element of when it was created. Also, the data is non-volatile in the sense that data is added and never removed, until the data is set for archiving. Whereas a complete data warehouse combines databases across an entire enterprise, data marts are usually smaller and focus on particular subjects, such as the case of this project. The project can present analysis information about the current Philippine Internet Map and AS connectivity. The use of the database enables the graphing tool to create maps that based on user’s specific time parameters unlike the previous approach that graphs the map based on the most current Rib and Update files such that there is no possibility of data comparison except a visual analysis. The graphing is independent on the insertion of the data so that analysis can be done concurrent with the insertion of the latest set of updates. However, information from the database is still five minutes delayed. A strictly real time graph cannot be derived because the Update Files come only after every five minutes.

Since the implementation is a Temporal Database, much information can be derived from it. One is the history of how the AS connections change from the time the database was started to the present timestamp. We are not limited anymore at looking the Philippine Internet Map at a specific time. Instead, the Map can be graphed based on different time and time spans.

Availability of ASes can also be derived based on how many disconnections they had.
Queries are also dynamic in such a way that users can change the time period they want to view the AS Connections.

Another advantage using this implementation is that Specific ASes or groups of ASes’ maps can be viewed separately from the entire Philippine Internet Map. This also gives the user more flexibility in gathering data.

Using a Temporal Relational Database, many things can still be done. In fact, the user has more freedom in creating the queries and processing that information.

The data in the database can be further viewed and analysed through the use of advanced Visualization tools. Currently, the neato and dot tools of GraphViz are limited only to drawing them in a 2-d space. The use of 3D visualization tools will make it more understandable and usable. Graphs drawing the evolution of the map can be done in order to show the changes that happened in a more animated way. And also, statistics about these changes can be studied. Data mining regarding the behaviour of the graph can also be done in the future.

Temporal Database has several access methods. These access methods use different kinds of data structures, most of them are tree-like, to make the search and update of the data more efficient and faster. The use of these temporal access methods data structure will improve the performance of the system.

Table schema can also be improved through the use of fields with low-level data structures. An example of this is using an integer instead of Strings in the prefix field. Additional information about the ASes, like address or geographical location, and other information, should also be added in order to add more usability to the system.

Internet route data are highly dependent on the peers of the server that collects the routes. Thus, peering directly to big ISPs, such as Mozcom, would significantly increase the Internet data routes.
References


Appendix

Appendix A. Testing

The testing phase of the project is divided into the data-insertion part and data-retrieval part. Each part is discussed in detail below.

Data insertion:

The first file to be tested is the Rib file (base file) that contains complete connectivity data for a given time period and will serve as the first batch of data to be inserted into the database. An input line is in the form:

Table Dump|Timestamp|Type|PeerIP|PeerAS|Prefix|ASPATH|Origin|NextHop|Local_Pref|MED|Community
i.e. TABLE_DUMP|1065696684|B|202.90.129.146|1|25.78.32.0|21|15 16 15|IGP|202.90.130.2|100||NAG|

The time is number of seconds since epoch when the packet was recorded. PeerIP and PeerAS are the IP address and AS number of the BGP peer from which we received the update. Prefix is the route prefix described in the update. ASPATH is the list of ASNs connected to the PeerAS, with the first ASN in the path connected to the PeerAS and the other ASNs connected to the preceding ASN in the said path. For the purposes of the project, we are only interested in obtaining the time, prefix, and ASPATH.

Type: Rib (base file)

Result: All pass!

<table>
<thead>
<tr>
<th>Case Number</th>
<th>Equal to Origin?</th>
<th>Within Brackets?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>1-2</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>2-1</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>2-2</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
### Number in AS Path: 2

<table>
<thead>
<tr>
<th>Case Number</th>
<th>Equal to Origin?</th>
<th>Within Brackets?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>No (ASes equal to each other)</td>
<td>No</td>
</tr>
<tr>
<td>1-2</td>
<td>No (ASes equal to each other)</td>
<td>Yes (2 ASes)</td>
</tr>
<tr>
<td>1-3</td>
<td>No (ASes equal to each other)</td>
<td>Yes (2 ASes)</td>
</tr>
<tr>
<td>1-4</td>
<td>No (ASes different from each other)</td>
<td>No</td>
</tr>
<tr>
<td>1-5</td>
<td>Yes (ASes equal to each other)</td>
<td>Yes (2 ASes)</td>
</tr>
<tr>
<td>1-6</td>
<td>Yes (ASes equal to each other)</td>
<td>Yes (2 ASes)</td>
</tr>
<tr>
<td>2-1</td>
<td>Yes (ASes equal to each other)</td>
<td>No</td>
</tr>
<tr>
<td>2-2</td>
<td>Yes (ASes equal to each other)</td>
<td>Yes (2 ASes)</td>
</tr>
<tr>
<td>2-3</td>
<td>Yes (ASes equal to each other)</td>
<td>Yes (2 ASes)</td>
</tr>
<tr>
<td>2-4</td>
<td>Yes (ASes different from each other)</td>
<td>No</td>
</tr>
<tr>
<td>2-5</td>
<td>Yes (ASes different from each other)</td>
<td>Yes (2 ASes)</td>
</tr>
<tr>
<td>2-6</td>
<td>Yes (ASes different from each other)</td>
<td>Yes (2 ASes)</td>
</tr>
</tbody>
</table>

### Number in AS Path: 3

<table>
<thead>
<tr>
<th>Case Number</th>
<th>ASes Equal to Origin?</th>
<th>Within Brackets?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>No (ASes equal to each other)</td>
<td>No</td>
</tr>
<tr>
<td>1-2</td>
<td>No (ASes equal to each other)</td>
<td>Yes (3 ASes)</td>
</tr>
<tr>
<td>1-3</td>
<td>No (ASes equal to each other)</td>
<td>Yes (2nd &amp; 3rd AS)</td>
</tr>
<tr>
<td>1-4</td>
<td>No (ASes equal to each other)</td>
<td>Yes (3 ASes)</td>
</tr>
<tr>
<td>2-1</td>
<td>No (1st and 2nd AS equal to each other)</td>
<td>No</td>
</tr>
<tr>
<td>2-2</td>
<td>No (1st and 2nd AS equal to each other)</td>
<td>Yes (3rd AS)</td>
</tr>
<tr>
<td>2-3</td>
<td>No (1st and 2nd AS equal to each other)</td>
<td>Yes (2nd &amp; 3rd AS)</td>
</tr>
<tr>
<td>2-4</td>
<td>No (1st and 2nd AS equal to each other)</td>
<td>Yes (3 ASes)</td>
</tr>
<tr>
<td>3-1</td>
<td>No (2nd and 3rd AS equal to each other)</td>
<td>No</td>
</tr>
<tr>
<td>3-2</td>
<td>No (2nd and 3rd AS equal to each other)</td>
<td>Yes (3rd AS)</td>
</tr>
<tr>
<td>3-3</td>
<td>No (2nd and 3rd AS equal to each other)</td>
<td>Yes (2nd &amp; 3rd AS)</td>
</tr>
<tr>
<td>3-4</td>
<td>No (2nd and 3rd AS equal to each other)</td>
<td>Yes (3 ASes)</td>
</tr>
<tr>
<td>4-1</td>
<td>No (1st and 3rd AS equal to each other)</td>
<td>No</td>
</tr>
<tr>
<td>4-2</td>
<td>No (1st and 3rd AS equal to each other)</td>
<td>Yes (3rd AS)</td>
</tr>
<tr>
<td>4-3</td>
<td>No (1st and 3rd AS equal to each other)</td>
<td>Yes (2nd &amp; 3rd AS)</td>
</tr>
<tr>
<td>4-4</td>
<td>No (1st and 3rd AS equal to each other)</td>
<td>Yes (3 ASes)</td>
</tr>
<tr>
<td>5-1</td>
<td>No (ASes different from each other)</td>
<td>No</td>
</tr>
<tr>
<td>5-2</td>
<td>No (ASes different from each other)</td>
<td>Yes (3rd AS)</td>
</tr>
<tr>
<td>5-3</td>
<td>No (ASes different from each other)</td>
<td>Yes (2nd &amp; 3rd AS)</td>
</tr>
<tr>
<td>5-4</td>
<td>No (ASes different from each other)</td>
<td>Yes (3 ASes)</td>
</tr>
<tr>
<td>6-1</td>
<td>Yes (ASes equal to each other)</td>
<td>No</td>
</tr>
<tr>
<td>6-2</td>
<td>Yes (ASes equal to each other)</td>
<td>Yes (3rd AS)</td>
</tr>
<tr>
<td>6-3</td>
<td>Yes (ASes equal to each other)</td>
<td>Yes (2nd &amp; 3rd AS)</td>
</tr>
<tr>
<td>6-4</td>
<td>Yes (ASes equal to each other)</td>
<td>Yes (3 ASes)</td>
</tr>
<tr>
<td>7-1</td>
<td>Yes (1st and 2nd AS equal to each other)</td>
<td>No</td>
</tr>
<tr>
<td>7-2</td>
<td>Yes (1st and 2nd AS equal to each other)</td>
<td>Yes (3rd AS)</td>
</tr>
<tr>
<td>7-3</td>
<td>Yes (1st and 2nd AS equal to each other)</td>
<td>Yes (2nd &amp; 3rd AS)</td>
</tr>
<tr>
<td>7-4</td>
<td>Yes (1st and 2nd AS equal to each other)</td>
<td>Yes (3 ASes)</td>
</tr>
<tr>
<td>8-1</td>
<td>Yes (2nd and 3rd AS equal to each other)</td>
<td>No</td>
</tr>
<tr>
<td>8-2</td>
<td>Yes (2nd and 3rd AS equal to each other)</td>
<td>Yes (3rd AS)</td>
</tr>
<tr>
<td>8-3</td>
<td>Yes (2nd and 3rd AS equal to each other)</td>
<td>Yes (2nd &amp; 3rd AS)</td>
</tr>
<tr>
<td>8-4</td>
<td>Yes (2nd and 3rd AS equal to each other)</td>
<td>Yes (3 ASes)</td>
</tr>
<tr>
<td>9-1</td>
<td>Yes (1st and 3rd AS equal to each other)</td>
<td>No</td>
</tr>
<tr>
<td>9-2</td>
<td>Yes (1st and 3rd AS equal to each other)</td>
<td>Yes (3rd AS)</td>
</tr>
<tr>
<td>9-3</td>
<td>Yes (1st and 3rd AS equal to each other)</td>
<td>Yes (2nd &amp; 3rd AS)</td>
</tr>
<tr>
<td>9-4</td>
<td>Yes (1st and 3rd AS equal to each other)</td>
<td>Yes (3 ASes)</td>
</tr>
<tr>
<td>10-1</td>
<td>Yes (ASes different from each other)</td>
<td>No</td>
</tr>
</tbody>
</table>
The second file to be tested is the Update file that will be the source of the succeeding batches of data to be inserted into the database. An input line is in the form:

Protocol|Timestamp|Type|PeerIP|PeerAS|Prefix|<update dependant information>
---|---|---|---|---|---|---
BGP4MP|1067789009|W|202.90.129.146|1|105.45.63.3/24
BGP4MP|1067789024|A|202.90.129.146|1|41.78.32.0/21|15|16|15|IGP|202.90.129.145|100|0||NAG|

The *timestamp* is number of seconds since epoch when the packet was recorded. The *type* is A for announcement or W for withdrawal. *PeerIP* and *PeerAS* are the IP address and AS number of the BGP peer from which we received the update. *Prefix* is the route prefix described in the update. For announcements, the *<update dependant information>* includes ASPath. For the purposes of the project, we are only interested in obtaining the time, ASPath (if applicable) and prefix.

**Type: Update File**

<table>
<thead>
<tr>
<th>Case Number</th>
<th>Description</th>
<th>Expected Outcome</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>Announce prefix (p-A) not in DB</td>
<td>p-A inserted into DB</td>
<td>Pass!</td>
</tr>
<tr>
<td></td>
<td>Withdraw prefix (p-B) not in DB</td>
<td>Do nothing</td>
<td>Pass!</td>
</tr>
<tr>
<td></td>
<td>Withdraw prefix (p-C) in DB</td>
<td>p-C deleted in DB</td>
<td>Pass!</td>
</tr>
<tr>
<td></td>
<td>Announce prefix (p-D) in DB</td>
<td>Do nothing</td>
<td>Pass!</td>
</tr>
<tr>
<td></td>
<td>Announce prefix (p-E) twice (p-E dne. in DB)</td>
<td>p-E not inserted into DB</td>
<td>Pass!</td>
</tr>
<tr>
<td></td>
<td>Announce prefix (p-F) twice, each with different time stamps (p-F dne. in DB)</td>
<td>p-F inserted only once</td>
<td>Pass!</td>
</tr>
<tr>
<td>2-1</td>
<td>Announce ASN1 with many prefixes</td>
<td>ASN1 inserted into DB</td>
<td>Pass!</td>
</tr>
<tr>
<td></td>
<td>Announce ASN2 with many prefixes</td>
<td>ASN2 inserted into DB</td>
<td>Pass!</td>
</tr>
<tr>
<td></td>
<td>Withdraw ASN1 (withdraw all prefixes)</td>
<td>ASN1 deleted from DB</td>
<td>Pass!</td>
</tr>
<tr>
<td></td>
<td>Withdraw ASN2 (withdraw all prefixes)</td>
<td>ASN2 deleted from DB</td>
<td>Pass!</td>
</tr>
<tr>
<td>3-1</td>
<td>Announce ASN1 with one prefix</td>
<td>ASN1 inserted into DB</td>
<td>Pass!</td>
</tr>
<tr>
<td></td>
<td>Announce ASN2 with many prefixes</td>
<td>ASN2 inserted into DB</td>
<td>Pass!</td>
</tr>
<tr>
<td></td>
<td>Withdraw ASN1 (withdraw prefix)</td>
<td>ASN1 deleted from DB</td>
<td>Pass!</td>
</tr>
<tr>
<td></td>
<td>Withdraw ASN2 (withdraw all prefixes)</td>
<td>ASN2 deleted from DB</td>
<td>Pass!</td>
</tr>
<tr>
<td>Announce ASN2 with many prefixes</td>
<td>ASN2 inserted into DB</td>
<td>Pass!</td>
<td></td>
</tr>
<tr>
<td>Announce ASN2 with same prefixes above</td>
<td></td>
<td>Pass!</td>
<td></td>
</tr>
</tbody>
</table>

Legend:
DB – Database
dne – does not exist in DB