A TRAFFIC DATA WAREHOUSING AND VISUALIZATION SCHEME

A THESIS

SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL

OF THE UNIVERSITY OF MINNESOTA

BY

DEODATTA BHOITE

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

JULY 2004

UNIVERSITY OF MINNESOTA

This is to certify that I have examined this copy of a master's thesis by

DEODATTA BHOITE

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Dr. Richard Maclin

Name of Faculty Adviser(s)

Signature of Faculty Adviser(s)

Date

GRADUATE SCHOOL

Acknowledgements

I would like to thank the Northland Advanced Transportation Research Systems Laboratory for providing the funding for this research. I would also like to thank the department of Computer Science at University of Minnesota, Duluth for providing the computers and facilities to make this work possible. This work would not have been possible without the guidance of my thesis advisor Dr. Richard Maclin and feedback from Dr. Carolyn Crouch and Dr. Donald Crouch. I would especially like to thank all the team members for their contribution and hard work towards completing this research, namely fellow graduate student Aniruddha Mahajan and undergraduate students Jeffrey Sharkey, Hemal Lal, Michael Perkins, Manish Rajkarnikar, Chun Ki Shin and Anthony Wilson. I would like to thank the system administrator Jim Luttinen who did a lot of work in reconfiguring the file system for the data, maintaining the database, backing up the database and maintaining of the database and web server. I am also grateful to the Traffic Management Center for collecting and archiving the traffic data that we use for this research. I am also thankful to Dr.Taek Kwon and the Transportation Data Research Laboratory for hosting the traffic data that we use for this research and making it publicly available. Lastly, I want to thank my family and my friends for supporting me and encouraging me to do my best.

Abstract

With the explosion of the amount of data available regarding transportation, such as traffic data, weather data, etc., it has become increasingly important that tools be developed to cope with this growing avalanche of data. In this work we present a spatio-temporal data warehouse to analyze the large amount of traffic data that is generated by inductive loop sensors placed on freeways across Minnesota.

The data warehouse we created is a novel construct that splits the space and time dimensions to facilitate the aggregation of data at various levels of abstraction. Our data warehouse has a web-based, user-friendly interface to perform ad-hoc queries and to visualize the results using techniques such as map visualization, bar graphs, heat maps and line graphs. These high-dimensional visualization techniques reveal interesting patterns in the data that otherwise might be hard to find manually. We also employ data mining algorithms for discovering association rules, performing clustering analysis and classifying the data. The combination of the data warehouse, visualization tools and data mining algorithms are shown to be useful for discovering interesting patterns in the data.

We believe that our solution will be helpful to traffic personnel and others interested in traffic data to better understand and analyze the enormous amount of data that is generated by traffic sensors and to utilize these analyses to solve critical traffic problems. We also believe that this work is a useful contribution to Intelligent Transportation Systems research.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

With the ever growing number of automobiles on our road system it is highly important to understand the flow of traffic in order to improve traffic conditions and travel time. For example, a person may want to find out how much time to budget so that he reaches work on time. Travel time estimation and traffic patterns can assist that person in deciding when he should leave for work. Understanding traffic flow problems and the ability to predict traffic conditions at a particular place and time can also help us take proactive actions to mitigate potential traffic jams. For example, using data mining we may notice that a certain part of the highway I-35 has significant traffic on Friday evenings at rush-hours. On investigation we find out that this bad traffic is due to people yielding for a left turn and stagnating the traffic behind. We can mitigate this situation by disallowing left turns at rush-hours in that section of I-35.

Intelligent Transportation Systems play a very important role in understanding traffic conditions and aid in making decisions related to construction of new highways and other measures to improve the traffic conditions. Such systems are either based on real-time traffic data or historical traffic data. Real-time data is used to determine immediate traffic

conditions and predict travel times based on the current situation. Historical data is used to determining long-term patterns in traffic conditions. The patterns can be used to predict traffic conditions. One technique that makes effective use of historical data to perform rapid analyses and make business decisions is a data warehouse.

Data warehouses contain historical traffic data along with visualization techniques and data mining tools and can help by determining traffic patterns, discovering anomalies, predicting travel times and traffic conditions and finding interesting relationships in the data. These results can help the commuter to understand the traffic and choose a route and time to avoid delays. The results also enable traffic authorities to take proactive actions to mitigate bad traffic conditions.

1.2 Thesis Statement

This research can be described as developing an intelligent transportation system that attempts to solve the problem of understanding traffic flow patterns and predicting traffic conditions within that system. We have followed the approach of creating a data warehouse of the historical traffic data from inductive loop sensors around the state of Minnesota and have built various visualization techniques and data mining tools that will make use of this data to discover interesting patterns and relationships in the data.

We created a data warehouse to effectively represent the traffic data. Our work includes a novel method for dividing the data along its natural dimensions. We found that it is easier to answer complex questions using this technique instead of the traditional approach for designing the data warehouse dimensions. For example, it is easier to answer a complicated question like "What is the average rush-hour traffic on weekends in summer on north bound highway I-35?" using our approach. We also show how this data warehouse can be used to perform ad-hoc queries for rapid analyses and to find novel patterns by using visualizations and data mining algorithms.

The visualization techniques present the traffic data in a manner that facilitates better comprehension of the enormous amount of data. We create several visualizations that help the user understand the tremendous amount of data and discover novel patterns in the data using a simple web-based tool. For example, a user may determine the average traffic on a certain highway at rush-hours on Friday evening using our visualization techniques and then compare them to the traffic conditions at the same time on an alternate route. This analysis will help him determine which route to use.

The visualization techniques are supplemented by the data mining tools that discover novel patterns in the data that can be used to predict traffic conditions. For example, association rules like "If the traffic is high on I-35 between 4pm to 8pm on Friday then it is also high on I-694 at the same time" may help a person in deciding to avoid highway I-694 if he is aware of the bad traffic conditions on I-35 on Friday evenings.

The data warehouse with visualization techniques and data mining tools together constitute a method for determining interesting patterns in the traffic data and for determining traffic conditions. We believe that this approach will effectively address the problems discussed in the previous section.

1.3 Thesis Outline

This thesis is organized as follows. Chapter 2 presents the background on data warehouses, the various data visualization techniques employed, and data mining algorithms implemented. The following chapter describes the factors that we considered while building the traffic data warehouse and the steps taken to implement it. It also covers the visualizations implemented to analyze the data set, and data mining algorithms implemented to reveal novel patterns in the traffic data. Chapter 4 demonstrates the usage of the data warehouse, visualization schemes and data mining algorithms. Chapter 5 discusses other work related to ours and delineates how our research differs from the current research on the problem.

The next chapter describes future work that could be done on the problem. Finally, we conclude the thesis by describing the main outcomes of the work.

Chapter 2

Background

This chapter will discuss the background for this research. In the first section we present data warehousing and related concepts. We will discuss the data cube model that is used in data warehousing and the primitives that are used to explore the data in that model. In the next section, we give details of the traffic data that was used for this research. Then we look into some techniques used for the visualization of high-dimensional spatio-temporal data. Finally, we elaborate on popular data mining techniques used to find interesting patterns in spatio-temporal data.

2.1 Data Warehouses

2.1.1 Introduction

With the explosion of the amount of data available relating to transportation, such as traffic data, weather data, road conditions, etc., it has become increasingly important that tools be developed to cope with this growing avalanche of data. A data warehouse is a tool used to organize and understand historical data and make decisions based on the knowledge discovered in that data.

For example, a store having many franchises will collect data from all its outlets regard-

ing sales. The data might detail the items that a customer bought and the time at which they were bought. The credit card or check used for the purchase can be used to discover information about the customer such as his or her income, age and occupation. This data is then stored in a central data warehouse to discover buying patterns related to variables such as age, occupation and income. This analysis can help the store do more business by doing targeted advertising based on the patterns found in the data, controlling their inventory by predicting the expected sales of an item and other business analysis tasks. For example, if a store finds out that men between the age of 25 and 30 with an income between $50,000 and $80,000 are more likely to buy computers, then they can send tempting offers by coupons in the mail to such customers to increase their computer sales.

Han and Kamber [HK01] define a *data warehouse* as

> *A semantically consistent data store that serves as a physical implementation of a decision support data model and stores the information on which an enterprise needs to make strategic decisions.*

Inmon [Inm96] defines a data warehouse as

> *A subject-oriented, integrated, time-variant, and non-volatile collection of data in support of management's decision making process.*

Both definitions indicate that the data warehouse is useful in a decision support systems (DSS). We will further expand on Inmon's definition which gives us four important properties of the data in a data warehouse.

The first property is that the data in the data warehouse has to be *subject-oriented.* In other words, before creating a data warehouse, the purpose of the creation of that data warehouse should be clear. For example, if a store wants to determine customer buying patterns, it should store its sales records rather that storing other information such as employee payroll in the data warehouse.

The second property of a data warehouse is that it should be *integrated*. Often, the data gathered for the data warehouse comes from heterogeneous sources, such as checkout machines in stores, relational databases, palm-tops of salespeople, etc. For example, a store might record some of its transactions in flat files and others in a relational database. It may also have salespersons that record their personal sales on palm-tops or on paper. All this data has to be entered into the data warehouse for a proper analysis of the sales of that store. There are several preprocessing tasks that we must carry out on this raw data before we enter it into the data warehouse. First, data must be cleaned. By this we mean that any inconsistent data should be removed. For example, if the data is gathered by a salesperson, there may be some errors in the data recorded like an incorrect date, an incorrect product identifier and so on. Such data should be corrected before entering it into the data warehouse, or if it cannot be corrected, then it should not be entered into the data warehouse because it may lead to false conclusions. We must then integrate this data, meaning that the data from the heterogeneous sources must be converted to a uniform format before entering it into a data warehouse.

The third property of a data warehouse is that it should be *time-variant*. The decision support systems make decisions based on past experiences and patterns. Thus, in order to make such decisions we have to store historical data. It may not be wise to draw conclusions from data gathered over a small period of time. For example, a store that sold lots of toys during holiday season should not decide to raise its toy inventory level due to a single month's sales data. Any pattern has more chances of being confirmed if we have a large amount of data over a large period of time.

The fourth property of a data warehouse is that it should be *non-volatile*. The data in the data warehouse is typically never removed, because the larger the amount of the historical data, the better we will be able to support our decisions. However it may be archived sometimes to save space.

2.1.2 Steps in Building a Data Warehouse

A *schema* for a data warehouse is the design and the relationships between the tables that are created in a relational database. The building of a data warehouse is a time-consuming process and has to be done with careful planning because of the importance of the task and also because the quantity of the data is so huge that it is not very simple to go back and change the schema once the data is loaded. Following are the steps that need to be taken while building a data warehouse:

1. The first task is to identify the purpose of the data warehouse. Whether we need the data warehouse for sales, inventory, personnel management or some other task. A data warehouse may also be used for a combination of any of these tasks.

2. Once we have identified the need for the data warehouse we must identify the sources of data and determine the data formats for those sources. If they are incompatible then tools must be developed to transform the data into a consistent format and cleaning must be performed to remove inconsistent data.

3. We must then determine appropriate dimensions and measures to suit the purpose that was finalized in step 1. A discussion of dimensions and measures can be found in Subsection 2.1.3.

4. The schema for the data warehouse must be designed to house the uniform data format that has been finalized in step 2. We must keep in mind what kind of queries we will run on the data warehouse while designing this schema to optimize the queries and also reduce the space required by the data. Examples of schemas include the star schema, the snowflake schema and the constellation schema. Details of these schemas can be found in Subsection 2.1.4.

5. The most important task for a data warehouse is to determine its granularity. The granularity is the detail in which the data is recorded in the data warehouse. If the

granularity is too high, then the size of the data warehouse will be too large, i.e., it will take up too much disk space. If the granularity is too low then we will not find any interesting patterns in the data and will lose many of the important details. Subsection 2.1.5 will discuss this issue in detail.

6. Once this is done then we must load the data into the data warehouse and optimize the data warehouse for the queries that we will be running on it. This will be discussed in Subsection 2.1.6. We will also develop tools for querying, analyzing the data for interesting patterns, visualizing the data and interactively exploring the data. Section 2.2 will discuss the visualization techniques and Section 2.3 will discuss the algorithms we use for analyzing this data.

2.1.3 Data Cube Model

A *data cube* is a model for storing multidimensional data that was introduced by Gray et al., [GCB+97]. It stores the data arranged by dimensions. A *dimension* is a variable along which data is measured. For example, the temperature can be measured at different times and at different places, so time and space form the dimensions for the temperature data.

To implement a data warehouse we need to identify the dimensions along which the data has to be stored and analyzed to find patterns. For example, if a store wants to find patterns of sales of various products during various times of year at various locations, then it would store the sales amount with respect to the dimensions time of the year, location and products. A fictional example of the data for such a set of stores is shown in Table 2.1.

The values stored at various points along the dimensions are known as *measures*. Examples of a measure are the amount of sales in dollars, the quantity of sales in terms of number of units sold and number of cars passing a highway. Figure 2.1 shows a sample data cube with three dimensions and total sales as the measure.

Usually the data in the data cube is summarized due to the large size of the data. For

Table 2.1: **Annual sales for a store.** This table shows fictional total sales in thousands of dollars for a set of stores according to dimensions location, time of the year (quarters) and product types (E for electronics, T for toys and C for consumables). Here the total sales in thousands of dollars for each store in each category during the various quarters is the measure used.

Location	Qtr 1			Qtr 2			Qtr 3			Qtr 4		
	E	T	C	E	T	C	E	T	C	E	T	C
New York	104	58	87	112	63	176	97	46	134	133	78	156
Chicago	123	72	111	102	48	156	84	122	85	134	103	133
Los Angeles	88	45	103	89	124	132	95	106	80	157	75	123
Detroit	93	78	67	69	85	88	68	71	91	84	35	45

example, instead of storing each of the millions of transactions that might have taken place during a particular quarter we summarize all transactions in that quarter and store only the total sales for that quarter in the data warehouse. The degree to which a dimension is summarized is known as the *granularity* and will be discussed in detail in Subsection 2.1.5.

The summarized data in the data cube can be analyzed for patterns using various operations on the data cube. Examples of the operations that can be done on the data cube include roll-up, drill-down, slice, dice and pivot. The roll-up operation reduces one or more dimensions and aggregates the values along remaining dimensions. For example, we can perform a roll-up on the product types and see the total sales of the store per quarter by location. The drill-down operation adds an additional dimension or divides a dimension into more parts to drill down the cube. For example, we can perform a drill-down on the time of the year dimension and summarize the sales per month to get the sales for a particular product type per location per month. In other words, a roll-up operation shows a summarized view of the data while the drill-down operation shows a more detailed view of the data. It may be useful to note that the operations roll-up and drill-down

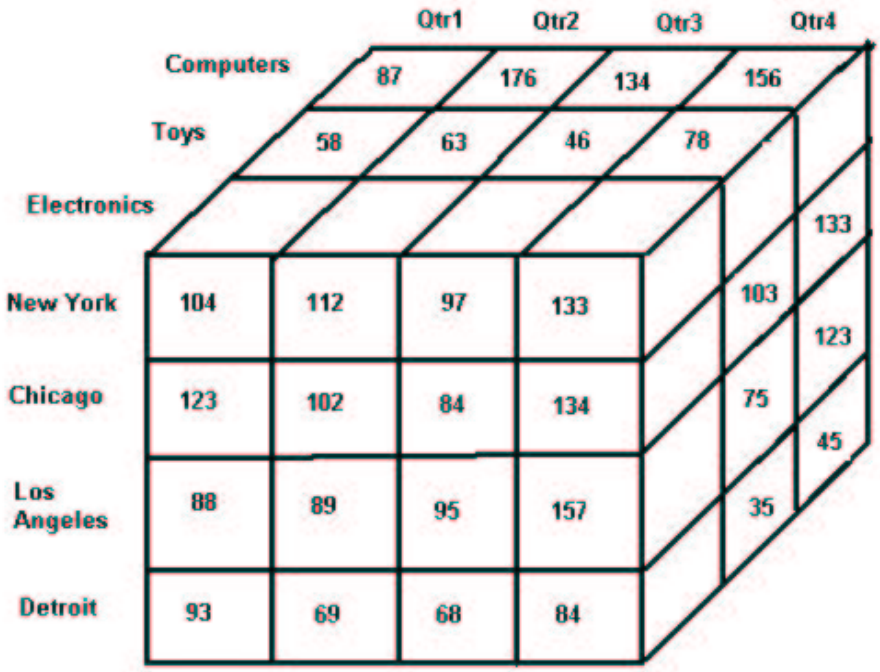

Figure 2.1: **Data cube for store sales.** The figure shows a sample data cube with three axes: (1) location; (2) time of year (in quarters); and (3) the type of products. Each value in the cube represents the amount of sales for the store in thousand dollars at the location for a particular quarter for a particular product type. (For example, the bottommost-left entry 93 tells us that the store in Detroit sold 93 thousand dollars worth of electronics in Quarter 1).

are possible when we have a concept hierarchy for the dimensions. For example, if we have stored the data along a time dimension that can be summarized by hour, day, week, month or year, then we cannot drill-down on the time dimension once we are looking at the data summarized by hour and we cannot further roll-up the time dimension if we are summarizing it by year. However, we may still perform a roll-up by eliminating the time-dimension. In the absence of summarized data we cannot perform a drill-down operation. The slice operation selects a subset of a dimension and dice operation selects a subset of two or more dimensions. For example, we may slice the location dimension to see sales of the New York store only and thus get the sales in New York outlet per product type per

quarter. A dice operation on the location dimension and the product type dimension can be performed to compare the sales of electronics items in New York outlet with those in Chicago outlet per quarter. The pivot operation is a visualization operation and it changes the point of view for observing the cube so we can look at it from a different perspective to discover different patterns.

2.1.4 Schemas for Multidimensional databases

Subsection 2.1.3 discussed the dimensions and measures required in a datacube. We model the dimensions and measures in the schema using dimension tables and a fact table. The dimension tables hold the data for the dimensions and the fact table holds the values of the measures for different combination of the dimensions. For example, in the store example our dimension tables will be location, time of year and product types. The fact table will contain different combinations of all these three dimensions and will also contain the corresponding measure, for example, total sales corresponding to those dimension values.

We can model this schema using three standard models for modeling multidimensional data. They are the star schema, snowflake schema and fact constellation schema, discussed below.

Star Schema: This is the most commonly used schema when there is a large central fact table containing the data and the dimension tables are connected to this table.

Consider the example in Figure 2.2, the central fact table Sales contains the total sales for the store and the dimension tables are connected to this table via foreign keys. Each row in the fact table will contain different combinations of the dimensions and the corresponding total sales value for those combinations. For example, there may be a row containing the location ID for location New York (1), time of the year ID for quarter 1 (1), and product type ID for electronics (1), and the corresponding total sales value (104) in thousands of dollars.

The advantage of this schema is that we can execute queries very fast due to the small

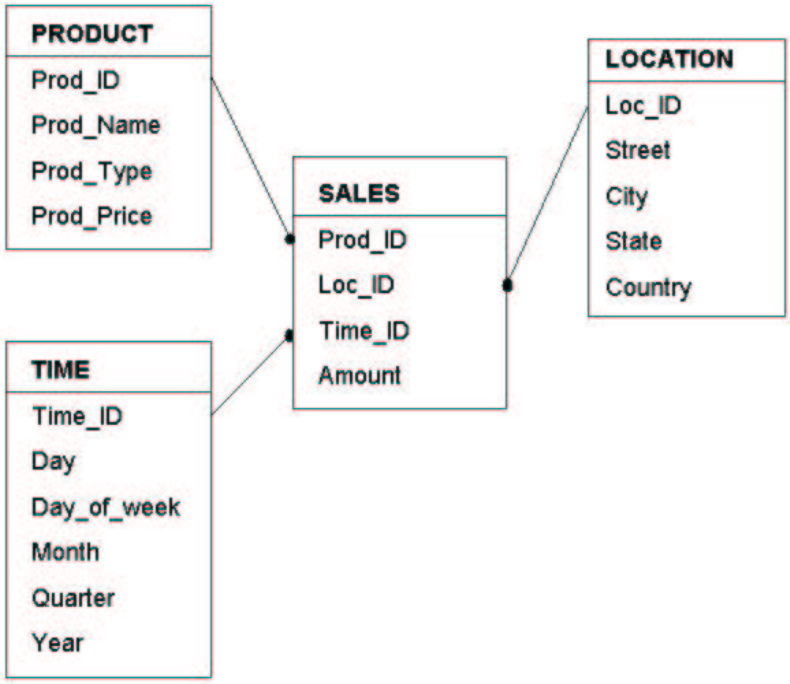

Figure 2.2: **Star Schema.** The data for the store from Figure 2.1 is organized in a star schema. There is a central fact table called SALES and the dimensions PRODUCT, LOCATION and TIME are connected to the fact table by foreign keys.

size of the dimension tables. The size of the fact table is smaller in this schema compared to storing the actual details of the dimensions in the fact table instead of keys to those details. For example, if we store the complete address of the store in the fact table instead of storing a location id in the fact table and storing the location details in a location table, then the size of the fact table will become very large.

The disadvantage of this technique is that the dimension tables are not normalized, which means that there is redundancy in the data. So the size of the dimension table may become large and the join between this dimension table and the fact table may become expensive.

Snowflake Schema: The snowflake schema is similar to the star schema except that it normalizes the big dimension tables by splitting them.

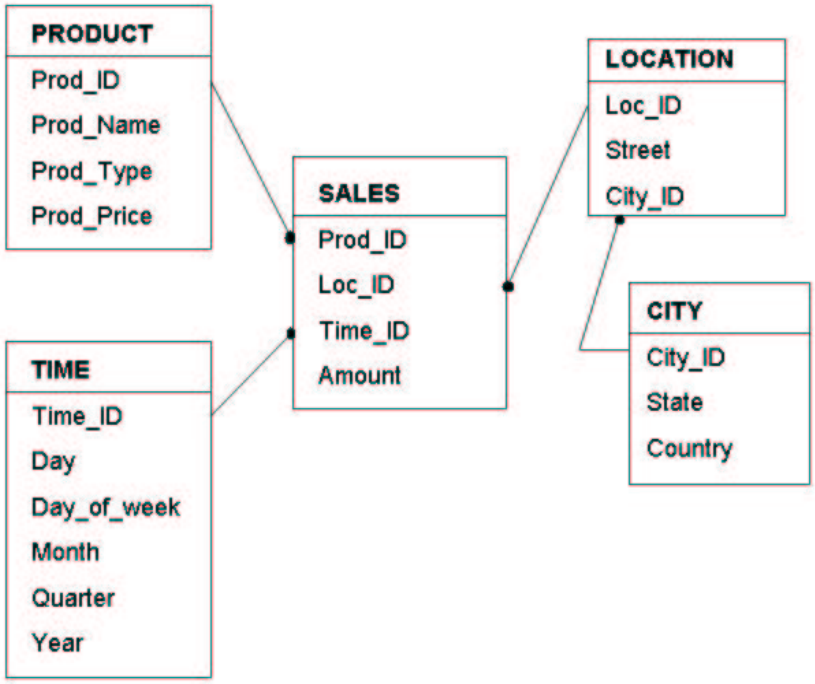

Figure 2.3: **Snowflake Schema.** The data for the store from Figure 2.1 is organized in a snowflake schema. This is similar to the star schema except that the big dimension table LOCATION is now split into two tables, one for location and the other for city.

Consider the example in Figure 2.3; the schema is similar to the star schema except that the location table is split into two tables since the store has a larger number of outlets in each city and it has also expanded to several other cities.

The advantage of this schema is that it saves storage space compared to the dimension tables in the star schema. However these savings are negligible compared to the size of the fact table.

The disadvantage of the snowflake schema is that we have to bear the overhead of joining more tables for the queries, so this schema is not as popular as the star schema.

Fact Constellation Schema: This is a more complicated schema that uses multiple fact tables which share the dimension tables. For example if the store also wants to keep track of its inventory and discover patterns that will help in improving its sales, it may also

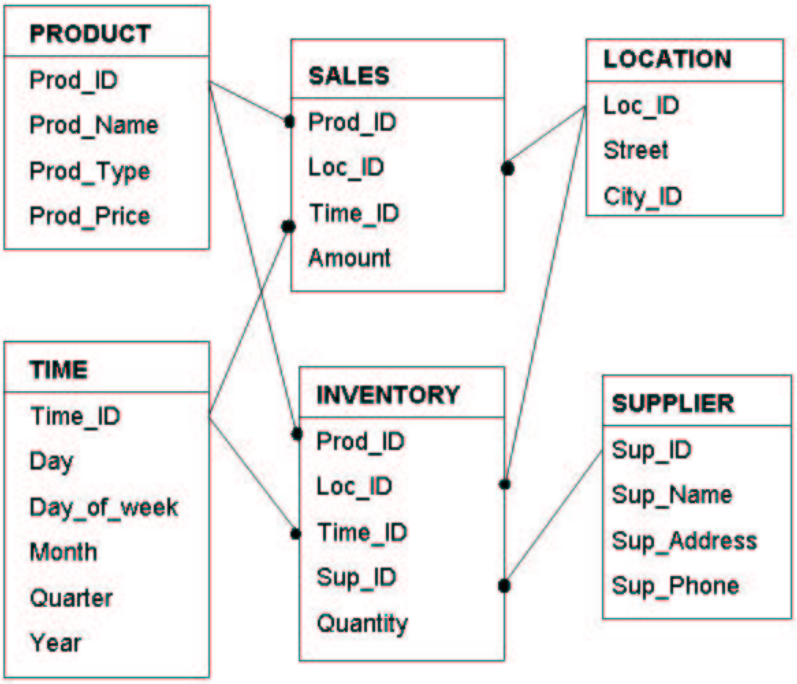

Figure 2.4: **Fact Constellation Schema.** The data for the store from Figure 2.1 is organized in a fact constellation schema. The store also needs to discover patterns in its inventory so it uses two fact tables SALES and INVENTORY. The dimension tables LOCATION, PRODUCT and TIME are shared by these two fact tables while the dimension table SUPPLIER is used exclusively for the INVENTORY fact table.

add a inventory fact table that will use location, product type and time of the year as its dimensions along with supplier as a dimension table. Figure 2.4 shows this schema.

The advantage of the fact constellation schema is the reuse of dimension tables to save storage space.

The disadvantages of this technique are the same as the star schema. The fact constellation schema may not be useful for small organizations due to its complexity.

2.1.5 Granularity of a Data Warehouse

Granularity is the amount of detail with which the data is stored in the data warehouse. It is important for a data warehouse to have the correct granularity for two reasons:

1. If the granularity is too high then the records are stored in too much detail. In this case, the size of the data warehouse will become very large and it will take a long time to query the data warehouse. So it will not be very useful for analysis.

2. If the granularity is too low then the records are summarized too coarsely. In this case, the data becomes very general and interesting information might be lost. Such a data warehouse is often not useful for analysis.

There is no simple way to determine the appropriate granularity for a data warehouse, but one can make estimates that will both preserve the interesting information in the data as well as keep size of the data warehouse manageable.

One way to estimate the granularity will be to determine the size of each row in the fact table and estimate the maximum and minimum number of rows that will be added to the fact table in a year. Usually 5 to 10 years of data is enough to make accurate predictions, though in other applications other time scales will be necessary. Once a set of possible granularities is determined, the size of the fact table can be calculated for each such combination. We can then pick a particular combination that is both manageable by our resources and will also provide a detailed view of the data for analysis.

Another approach is to use a data warehouse with dual granularity in which the recent data is kept in more detail and for varying lengths of time in greater or lesser detail. For example, we may store the 5 year old data for the store from Figure 2.1 summarized by every hour and store the 5 to 10 year old data summarized by each day. This technique is useful for saving space because the store may not need the older data summarized by every hour for analysis.

2.1.6 Optimizing the Data Warehouse

The size of the data warehouse and the complexity of queries often cause queries to take a long time to complete. There are several methods to improve the query speed.

Creating bit-map and join indexes is one of the ways queries can be optimized. We create bit-map indexes on columns that have low cardinality. Cardinality is the number of possible values that a field in the table can have. For example, if the possible number of values that a column can contain is less than 10, i.e., the cardinality of the field is less than 10, then we should prefer creating a bit-map index on the column to creating a normal index. A normal index uses a b-tree structure to make the key search more efficient whereas the bit-map index will only set the bits to store the same data. For example, if a column has two values then the normal index will use these actual values to store the data while the bit-map index will just use a single bit and set it or unset it to represent these two values. This technique makes the table scans faster and searching more efficient than the b-tree index.

Often we have to join the fact table and dimension tables and there can be multiple such joins in one query. When the dimension table in the join is also large then the join can be very expensive. To optimize this join we can create join indexes which reduce the time required to join two big tables and thereby improves the query timing.

Another way to optimize queries is to materialize the possible data cubes or frequently queried data cubes. By materialization we mean that we create materialized views of subsets of the data cube. Materialization of all possible data cubes is a very expensive operation and requires a huge amount of memory. Harinarayan et al., [HRU96] suggest methods for efficient materialization of data cubes, while Mumick et al., [MGM97] address the issues of maintaining the summary tables efficiently.

2.1.7 Data Warehouse Architecture

A typical data warehouse architecture is shown in Figure 2.5. Here we have shown that a data warehouse is often made up of three tiers.

The bottommost tier is the data warehouse tier. We preprocess the data and load it into the data warehouse for querying.

The middle tier is the Online Analytical Processing (OLAP) server tier. This layer processes the multidimensional operations on the data warehouse. These operations can either be done using Relational OLAP (ROLAP) or using a specialized server that implements multidimensional constructs, known as Multidimensional OLAP (MOLAP). This layer presents a multidimensional view of the data warehouse to the application layer. We discuss the OLAP technology in detail in Subsection 2.1.8.

The topmost tier is the application layer. This layer uses the multidimensional view of the data warehouse to perform analysis tasks, data mining or visualization tasks. The user selects the task that he wants to perform on the data. This layer then translates this request into operations on the multidimensional data and sends them to the middle layer. The middle layer translates these multidimensional operations into structured query language (SQL) queries and returns the results to the application layer. The application layer then performs further processing on this data and presents the results to the user.

For example, a company can create a data warehouse from the sales data that it generates which forms the bottom layer. It may use a relational database to store the data and use servlets to translate the multidimensional operations into SQL queries as its middle layer. In our work, the web-based applications form the topmost layer of the data warehouse that will provide data mining tasks to the users. The user employs the web-based tools to mine the sales data to find novel patterns. The web-based tool sends these requests to the servlets. The servlets translate these requests to SQL queries and query the data warehouse. The servlet sends back the data to the web-based tool that will further process

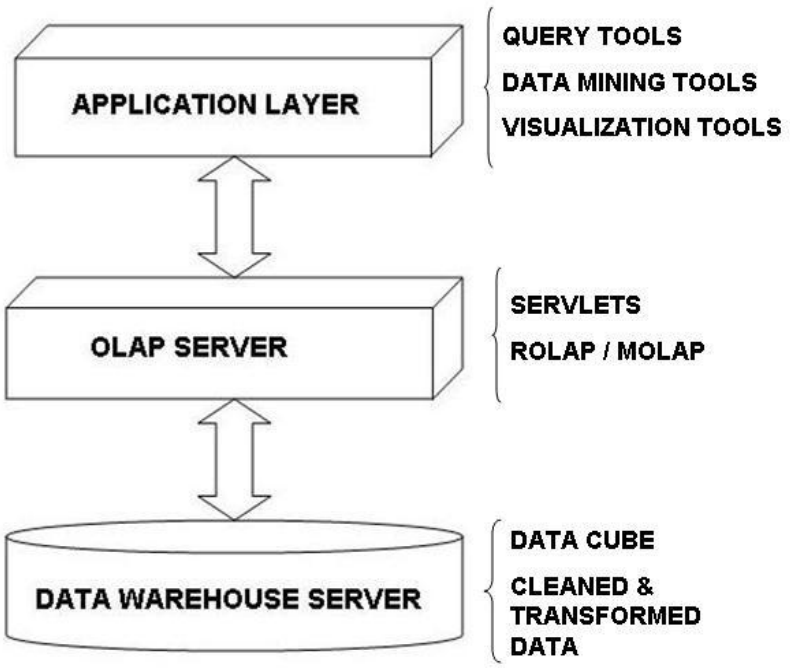

Figure 2.5: **Data Warehouse Architecture.** This figure shows the architecture of a typical data warehouse. There are 3 tiers to a data warehouse: (1) the data warehouse server that contains the cleaned and transformed data stored in a data cube structure; (2) the OLAP server that may be implemented by ROLAP or MOLAP servers or using Servlets; and (3) the application layer that may contain query tools, data mining tools and visualization tools.

the data and display the results to the user.

2.1.8 Online Analytical Processing Systems

Decision support systems associated with a data warehouses are sometimes known as On-line Analytical Processing (OLAP) systems. Systems used to store daily transactions and records are known as On-line Transaction Processing (OLTP) systems. The most important difference between OLTP and OLAP systems is that the data used for OLAP is summarized whereas that used for OLTP is detailed. The reason behind this is very straightforward; OLAP systems are used to find patterns in historical data, so we need not know the exact

value of each and every data point in the history, since that would amount to a gigantic database. For example, a company may use a database to log its daily transactions, however we may not be able to answer questions like the historical trends of the store from this data, since OLTP systems generally do not keep track of all historical data. OLAP systems store historical data for the company and will not store all transactions for each day, but will summarize those transactions into total daily sales or total weekly sales.

The other difference between OLAP and OLTP is that OLAP systems often perform only select queries on the data and not update or delete, but OLTP performs all kinds of queries on the data. The data in OLAP systems is typically never destroyed but OLTP systems will typically delete the old data otherwise the database would become unmanageably large.

2.2 Data Visualization

Visualizing high-dimensional data is always a challenge. For example, if you wanted to show the total sales with respect to a location you can easily show it as a bar graph. If you wanted to see the total sales per product type for each location you can use a colored bar graph or a stacked bar graph to represent the various product types and plot locations as the X-axis and total sales on the Y-axis. But it is very tricky to show the quarterly sales for each location for each product type. You could use multiple bar graphs or emulate 3 dimensions on the 2-dimensional surface by using a perspective view to show a 3-D bar graph. But it is still difficult to go on adding dimensions indefinitely. Visualizing more than 4 dimensions is a challenging task. Even if you are able to create a visualization scheme for such a data there is no guarantee that it will make sense to the user.

There are several properties that the good visualization should have:

1. It should be able to represent the data correctly and completely.

2. It should be easy and intuitive enough to allow the user to comprehend the data and

discover patterns in it.

3. It should be extensible to a larger number of dimensions if there is a need in the future to add more dimensions.

4. The user should be able to customize various aspects of the visualization like the line style, color, dimensions on various axes.

Apart from the high-dimensionality challenge we are also faced with a challenge of representing both space and time together in a visualization. In our system, described in Chapter 3, we implemented several visualizations so that a user may choose a visualization that he is comfortable with. One advantage of having more than one visualization is that the results from more than one visualization can be collated to find novel patterns in the data. For example, a user may find the outliers using the line visualization and find the common patterns using the bar graph visualization. Then the user may use the outliers and patterns in the traffic to gain more knowledge about the general traffic conditions and also may find whether the outliers occur periodically so as to predict the traffic conditions accurately.

In our system we used a map visualization to represent the spatial dimension and animated the visualization with respect to time to represent the various values for the time dimension. We also displayed results using bar graphs and heat maps so that the user is able to compare more than one point in time simultaneously. We will discuss these techniques in detail in Chapter 3.

2.3 Data Mining

Data Mining [HK01][HMS00] is the process of extracting meaningful information from vast quantities of data. There are several methods of extracting such information from the data.

Association Analysis is a class of techniques that help us associate multiple attributes of the data with the values, thus forming attribute-value pairs. It is highly important to find interesting relationships in the large amounts of data produced every day. For example, it may be useful for a superstore to find which items are bought together by a customer. The results of mining the data containing the buying habits of customers may reveal that males aged 30 to 35 having income between 60,000 and 80,000 dollars are likely to buy laptops. The superstore can send enticing deals to people with these attributes by mail which may result in increasing their revenues. Most of the "target" advertising is based on finding such interesting relationships.

Association analysis is generally unsupervised and is based on finding strong relationships from the frequent itemsets. There are usually two terms that are used in association analysis, namely *support* and *confidence*. Support is the proportion of the dataset satisfying a certain rule and confidence is the proportion of the dataset having those attributes that satisfy the rule. For example, the support for the rule {male, 30-35, 60,000-80,000} $\Rightarrow$ buys-laptop is 4% means that 4% of the dataset are males between the ages of 30 and 35 with incomes in the range 60,000 to 80,000 dollars. Confidence for this rule is 80% means that 80% of the males between age 30 to 35 and having an income between 60,000 and 80,000 dollars who shopped at this superstore have bought laptops.

An example of algorithm that performs association analysis is the Apriori algorithm [AIS93] that will be discussed in detail in Subsection 3.4.2. The Apriori algorithm discovers the interesting association rules in the data that satisfy the minimum support threshold and minimum confidence threshold.

Data characterization is a class of algorithms that allows us to characterize the features of the data based on a user-specified query. A simple example of a user-specified query for an electronics store's data might be to find the general characteristics of frequent buyers. The results of the mining algorithm could be that customers who purchase HDTVs are females between age 25 and 30 having a household income of more than 80,000 dollars.

We used Self Organizing Maps [Koh82][Koh90] for data characterization. The details of this algorithm are discussed in Subsection 3.4.1. Self Organizing Maps is an example of unsupervised algorithm that characterizes the data. It can also be used as a dimension reduction algorithm.

Clustering Analysis is an example of unsupervised analysis that is used to group related data into clusters. The user can find related features of this data and perform analysis using these clusters. Clustering algorithms are usually used in the absense of labeled data. For example, we may use clustering algorithms to determine related genes in a human chromosome. Since we do not know which genes will be functionally related we can perform a clustering analysis to find the related genes. Clustering analysis is discussed in detail in Subsection 3.4.3. K-means clustering is an example of an unsupervised algorithm that performs clustering analysis and determines the clusters in the data.

We will discuss the details of the data mining tasks that we performed on this data in Chapter 3.

Chapter 3

A Traffic Data Warehouse and Visualization

We implemented a data warehouse to store the traffic data that is collected by Traffic Management Center (TMC). TMC is a division of Minnesota Department of Transportation (MNDoT). This data is hosted by Dr. Taek Kwon's Traffic Data Research Laboratory [TDR02].In this chapter we present our system. We first discuss the traffic data used for this research. Then we will describe how the data warehouse was constructed for the data. We further elaborate on the visualization techniques used to visualize this data and also discuss the various data mining algorithms that are applied to this data.

3.1 Traffic Data Used for Research

The data used for this research is generated by 4,972 inductive loop sensors on the freeway system in Minnesota. Most of the sensors are located in the Twin Cities area. Each of the sensors takes two types of readings of the traffic every 30 seconds. The collected data is then packaged daily into a single zip file that is archived and made publicly available by Dr. Taek Kwon [KDPK03].

The sensor provides two types of readings:

1. **Volume:** The number of vehicles passing the sensor every 30 seconds.

2. **Occupancy:** The time during which the sensor was occupied within the 30 second interval.

Valid data in the volume files ranges from 0 to 40 since not more than 40 vehicles can be registered as passing a sensor in 30 seconds. Valid data in the occupancy files ranges either from 0 to 1800 or from 0 to 1000 depending on how it is formatted (c30 or o30). The c30 file format divides the 30 second interval into 1800 equal parts, whereas the older, o30 file format divides the 30 second interval into 1000 equal parts.

Volume data can be represented in one byte whereas occupancy requires two bytes every 30 seconds. Thus a file containing 24 hours traffic data for a sensor contains 2880 bytes of volume data and 5760 bytes of occupancy data per sensor.

Traffic data is available from the year 1999 to the current date. But many new sensors were added during that period so not every sensor has data for all five years. Since sensors malfunction from time to time, data can be missing or invalid. The yearly growth rate of this data is about 15.5GB. Over five years we have a total of about 77.5GB data.

To illustrate the differences between volume and occupancy, consider the following four conditions:

- **Low Volume, Low Occupancy** - very few cars are passing a sensor and the sensor has cars above it for only a short period of time. This would be an ideal time to travel.

- **Low Volume, High Occupancy** - in this condition, very few cars are passing but a car is occupying the sensor fairly often. This situation occurs when traffic is moving very slow and backs up. This condition may well correspond to a traffic jam.

- **High Volume, Low Occupancy** - in this condition, many cars are passing but occupancy remains low. Thus traffic is likely to be moving very well with little congestion.

- **High Volume, High Occupancy** - many cars are passing a sensor and there are cars over the sensor fairly often. This would suggest that the road is congested but that traffic is still moving.

The sensors are located on various highways in the state of Minnesota, primarily in the Twin Cities area. They cover major highways and some of the major connecting roads (such as I-35, I-94, I-694, I-494, I-35E, I35W, etc.).

We have 5 years of data for 4972 sensors. The total number of data points available for each sensor is approximately:

5 years * 365 days * 24 hours * 120 (half minutes) * 2 (measures) =

10,512,000 data points.

For 4972 sensors, this means up to $4.2X10^{10}$ data points. And of course the data continues to grow daily. Although disk space is relatively inexpensive, the amount of storage needed to hold this data and the time it would take to process it makes direct storage infeasible at this time. Thus we chose the approach of using a data warehouse, since many interesting questions can be answered from summarized data.

3.2 Traffic Data Warehouse

The traffic data warehouse was built for the purpose of performing rapid analyses on the traffic data and finding interesting patterns. The traffic data can be summarized over various dimensions for querying the data in ways that will provide different views of the data. Our data warehouse implementation will provide the means for storing and querying historical traffic data.

3.2.1 Determining the Dimensions and Measures

The traffic data has two dimensions. One is spatial, i.e., the location of the sensor, highway, etc. and the other is temporal, i.e., the time of the day, year, month, etc. Traditionally, we would have used a concept hierarchy for these two dimensions to model this data, however it is difficult to compare the rush hour and non-rush hour traffic for weekdays and weekends across all sensors, because it involves comparing two different levels in the concept hierarchy for a single dimension. Splitting these two dimensions into different dimensions allows us to perform such complex queries on the data.

We chose the following set of temporal dimensions:

1. **Time of the day:** This divides the traffic data across the 24 hours of the day. For example, we will be able to summarize the traffic data by rush hour and non rush hour.

2. **Day of the week:** This divides the traffic data across the seven days of the week. For example, we will be able to summarize the traffic data by weekends and weekdays.

3. **Time of year:** This divides the traffic data across the 12 months of the year. We will for example, be able to summarize the traffic data by the four seasons to find seasonal trends in the traffic.

4. **Year:** This divides the traffic data across years.

The spatial dimensions include:

1. **Sensor :** This divides the traffic data by sensor number. We will for example, be able to summarize the traffic data by different highways.

2. **Location :** This divides the traffic data by sensor location. We will for example, be able to summarize the traffic data by latitude and longitude.

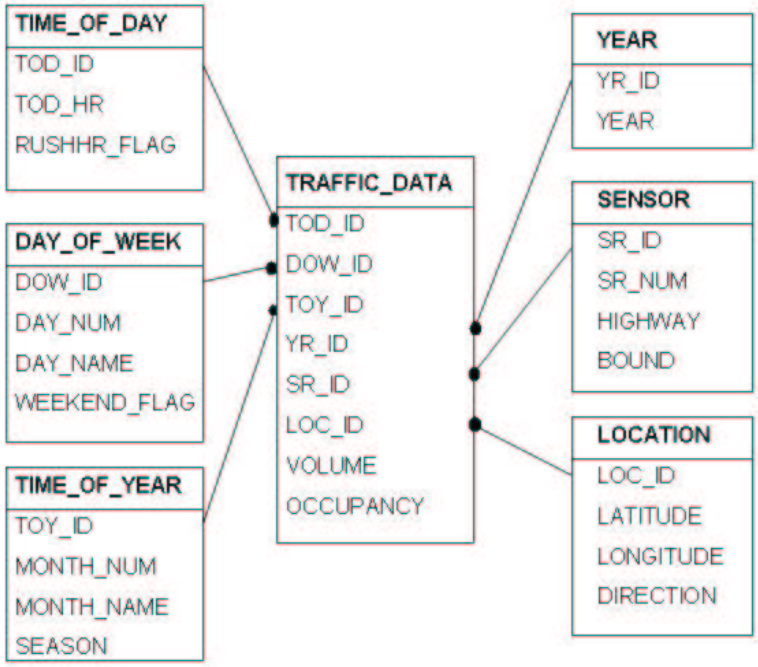

Figure 3.1: **Schema for the traffic data warehouse.** We used the star schema to model this data because none of the dimension tables were very large. The table TRAFFIC_DATA is the fact table for this schema and the remaining tables form the dimensions for this schema.

The choice of measures for this data was obvious. We used both volume and occupancy as the measures for this data warehouse in order to find trends in the traffic flow.

Since none of the dimension tables are very large we used the star schema to model the data. The actual schema is shown in Figure 3.1.

3.2.2 Determining the Granularity

After the schema was finalized we had to decide the level to which we will summarize the data. The detail with which we summarize the data is very important because if have too much detail about the data then the size of the database will become too large. If we have very little detail for example, if we summarize the data at a very high level then we lose

Table 3.1: **Choices of Granularity for dimensions** The table shows the different choices for the granularity of dimensions. All the dimensions have obvious choices for cardinality except time of the day where we would like to experiment with the different amount of time period over which traffic varies.

Choices	Time of Day	Day of Week	Month	Year	Sensor	Rows (million)
C1	8	7	12	5	4972	16.7
C2	24	7	12	5	4972	50.1
C3	48	7	12	5	4972	100.2
C4	288	7	12	5	4972	601.4

the interesting information in the data. We followed the empirical approach delineated in Subsection 2.1.5 and determined the possible combinations of granularity that we could use and found out the growth rate of the data. Table 3.1 shows the different combinations of granularity for the dimensions that we considered. We chose to use the second combination (C2) for our implementation because we can meaningfully get the traffic per hour and at the same time maintain a reasonable size for the database. We also implemented the last combination (C4) in a separate data warehouse. We decided to use this data as a backup if we do not find enough interesting patterns in the data summarized per hour. However, in our work we did not use the five minute summarized data warehouse.

3.2.3 Choice for Data warehouse software

We focussed our development on $MySQL^{TM}$ [MyS99b]. One of the biggest advantages of working with MySQL was that it has a wide range of data types like SHORT INT, TINY INT, etc. which when used in the fact table saves a lot of space. There are several benchmark tests [MyS99a] which indicate that MySQL is superior to other database softwares. It is also free to download and use.

3.2.4 Data Preprocessing and Loading

The traffic data we use was made available by Dr.Taek Kwon and his laboratory [TDR02]. It is archived using the zip utility. We created a script to unpack this data automatically. Once the data is unpacked we run a program to summarize the data by every hour and load it into the database.

If the volume is not between 0 and 40 or the occupancy is not between 0 and 1800 then categorize it as bad data and replaced by a marker indicating invalid data.

3.2.5 Data Warehouse Optimization

We built indexes on the tables to optimize them. First we found out how many records were being accessed by a query. This can be found out by doing an EXPLAIN on the query. For example all the records from the fact table (TRAFFIC_DATA) were being accessed by the following query.

```
mysql> select SR_ID, DOW_ID, AVG(VOLUME), AVG(OCCUPANCY)
    -> from TRAFFIC_DATA
    -> WHERE YR_ID=4
    -> AND (SR_ID=305 OR SR_ID=306)
    -> GROUP BY SR_ID, DOW_ID;
```

The query should have just accessed the records having sensor ID 305 or 306. In order to do that we create an index on SR_ID column in the TRAFFIC_DATA table.Optimizations such as this caused the query time to reduce drastically from an average of 52 seconds to 0.28 seconds. We created indexes on all the foreign keys in the TRAFFIC_DATA table to make the queries more efficient. A very complicated query might take a few minutes to execute in the current data warehouse, but most of them will execute under a minute.

3.3 Traffic Data Visualization

There were several considerations for the visualization schemes to use because we had to keep them simple, user-friendly and at the same time had to display high dimensional spatio-temporal data in such a way that it will help users find interesting patterns in the data.

We implemented several methods namely map visualization, bar graph visualization, heat map visualization and line graph visualization. We also provided several options that allow the user to customize those visualizations. The warehouse visualizer is designed to work as a web-based interface as well as a stand-alone application. The design allows the addition of many more visualizations without having to modify the existing code very much.

3.3.1 Map Visualization

Map visualization displays the spatial aspect of the data on a map by displaying the sensors on the highways and displaying the volume and occupancy by color, line thickness or dash frequencies. The temporal aspect is displayed by animating the map display. The user can play and pause the animation and can also look at specific time points by using the slider control for the visualization.

The advantage of the map visualization is that the user can easily find spatial patterns and gets a feel of how the traffic progresses over the time by the animation. The drawback of this visualization is that the user is unable to simultaneously "see" the traffic at two different times.

Figure 3.2 shows an example of map visualization.

3.3.2 Bar Graph Visualization

Bar graph visualization displays the volume and occupancy for different combinations of dimensions as a bar graph. We can have any number of dimensions on the X and Y axis

Figure 3.2: **Map Visualization.** This figure shows an example of map visualization. It shows the traffic on the sensors on northbound trunk highway number 212 on Friday evenings of January, 2002. The volume between the sensors is represented by the thickness of the line and the occupancy is represented by the color. Red color indicates high occupancy and yellow color indicates moderate occupancy. This visualization technique gives spatial information about the traffic to allow better comprehension of traffic flow on highways.

of the bar graph and we can also change the order in which they will be displayed. This configuration helps us to get different views of the query results to find out more interesting patterns in the data. We also provide an option of reflecting the bar graph across the X axis to display them as a curve. This arrangement makes it easier for the user to find patterns in the data across the Y axis.

The advantages of this visualization is that we can discover patterns across unrelated data items in a very easy way. However we lose the spatial aspect of the data and are not able to correlate between the patterns among spatially close sensors unless we are familiar with their locations.

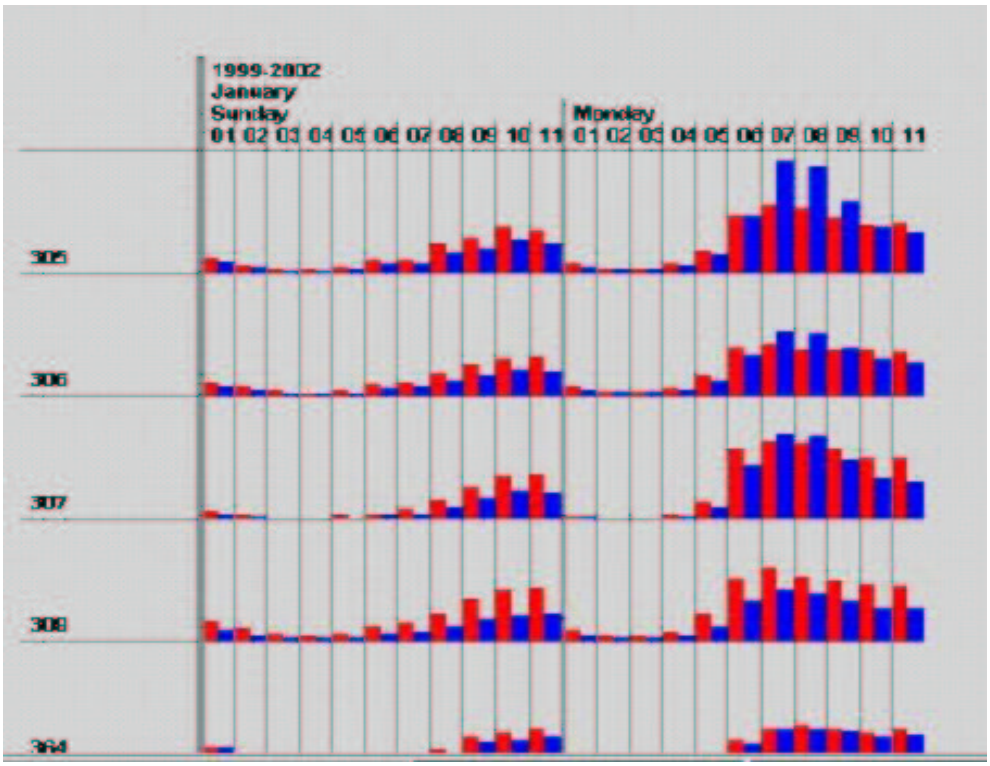

Figure 3.3: **Bar Graph Visualization.** This figure shows an example of bar graph visualization. The bar graph shows the average traffic on the sensors on northbound trunk highway 212 of Sunday and Monday in January between years 1999 to 2002 between midnight and noon. Using this visualization we can compare the traffic difference between Sunday and Monday. We note that the rush hour begins very late on Sunday as compared to Monday. It is easy to compare using this visualization technique.

Figure 3.3 shows an example of bar graph visualization.

3.3.3 Heat Map Visualization

The heat map visualization is similar to the bar graph visualization except that instead of displaying the volume and occupancy as bars we display them in a color coded fashion in boxes. The colors are decided by binning the volume and occupancy values. For example, the values of volume between 30-40 indicate a high volume and can be represented by red color and the values of volume between 0-10 indicate a low volume and can be represented by green color. This technique helps in finding more general patterns by looking for similar

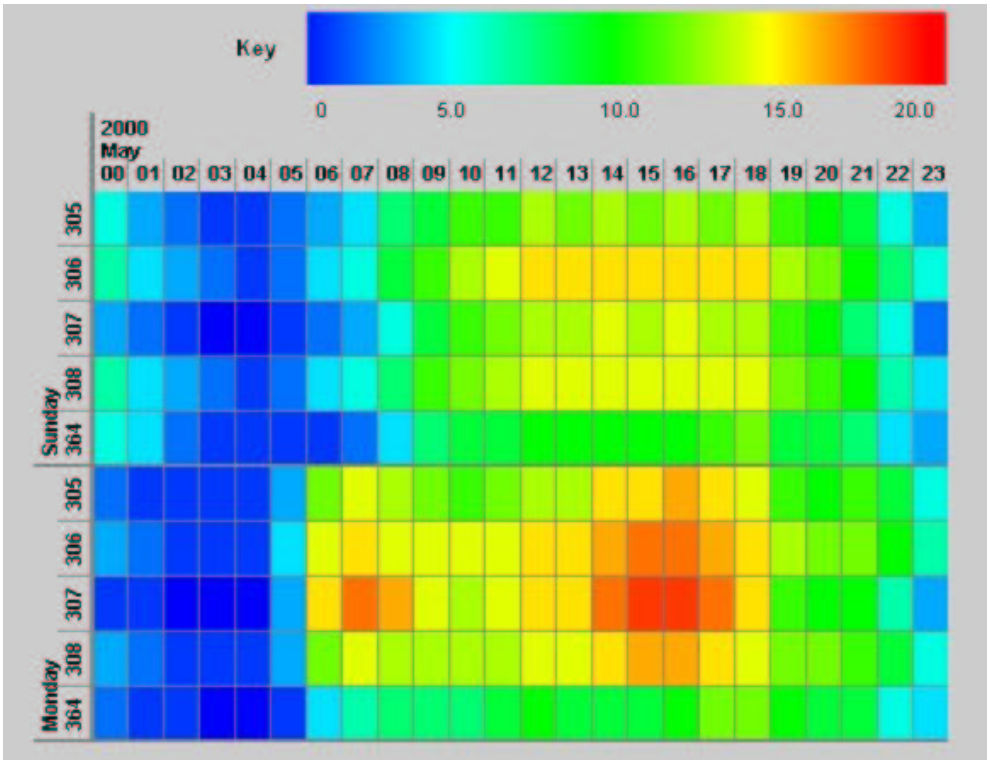

Figure 3.4: **Heat Map Visualization.** This figure shows an example of heat map visualization. It shows the average hourly traffic for May 2000 on northbound trunk highway 212 for Sunday and Monday. We note the two red spots in the lower region that indicate the morning and evening rush hour on Monday. The absence of such red spots indicate that there is no particular rush hour on Sunday. The blue region indicates the low traffic at late night. It is easy to find patterns using this visualization method.

colors or sharp changes in color values. It is very helpful in determining the actual rush hours for a particular area.

The advantage of the heat map visualization over the bar graph visualization is that we are able to find more general pattern in an easier way. The drawback is similar to that for the bar graph visualization; we lose the spatial coherence between two data points unless we are familiar with the location.

Figure 3.4 shows an example of heat map visualization.

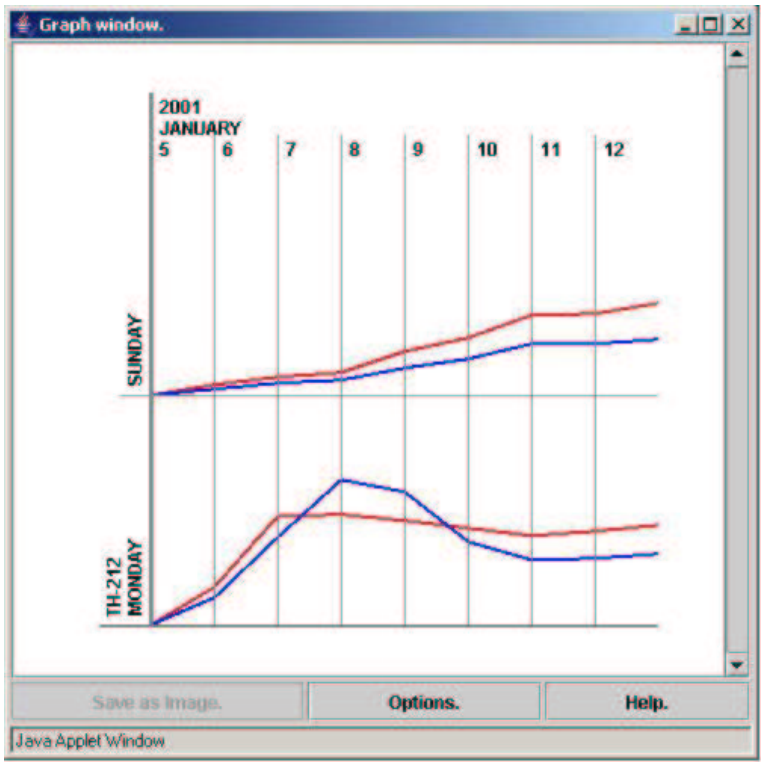

Figure 3.5: **Line Graph Visualization.** This figure shows an example of line graph visualization. It shows the average morning traffic on trunk highway 212 of January 2001. We note the slow and steady rise of traffic on Sunday as compared to the morning rush-hour peaks on Monday. It is easy to find outliers using this visualization technique.

3.3.4 Line Graph Visualization

Line graph visualization is similar to the bar graph visualization except that we display the volume and occupancy as connected lines instead of bars. This helps in comparing two data points on the Y axis in order to find similar patterns. The drawback of this method is same as for bar graph visualization.

An example of line graph visualization is shown in Figure 3.5.

3.3.5 XML Output

XML plays an important role in standardizing the language for communication between different organizations. By defining a data type definition (DTD) for a data and standardizing

it, we can use it to share data with different organizations or input it to various applications. It is very important for a database application to have XML parsing capability.

Following is the DTD that we created for this traffic data:

```
<?xml version="1.0"?>
<!DOCTYPE TRAFFIC_OUT [

    <!ELEMENT TRAFFIC_OUT (OUTPUT_ROW+)>

    <!ELEMENT OUTPUT_ROW (SENSORS?,LOCATIONS?,YEARS?,TIMES_OF_YEAR?,
  DAYS_OF_WEEK?, TIMES_OF_DAY?,(VOLUME|OCCUPANCY),
  (VOLUME|OCCUPANCY)?)>

    <!ELEMENT SENSORS (SENSOR+)>
    <!ELEMENT SENSOR (#CDATA)>

    <!ELEMENT LOCATIONS (LOCATION+)>
    <!ELEMENT LOCATION (#CDATA)>

    <!ELEMENT YEARS (YEAR+)>
    <!ELEMENT YEAR (#CDATA)>

    <!ELEMENT TIMES_OF_YEAR (TIME_OF_YEAR+)>
    <!ELEMENT TIME_OF_YEAR (#CDATA)>

    <!ELEMENT DAYS_OF_WEEK (DAY_OF_WEEK+)>
    <!ELEMENT DAY_OF_WEEK (#CDATA)>

    <!ELEMENT TIMES_OF_DAY (TIME_OF_DAY+)>
    <!ELEMENT TIME_OF_DAY (#CDATA)>

    <!ELEMENT VOLUME (#CDATA)>
    <!ELEMENT OCCUPANCY (#CDATA)>
]>
```

We incorporated this feature to allow the users to output the data in XML format and use their own applications to display or analyze the data using the DTD that we have created for this data and this XML output for the query the user executed.

Figure 3.6 shows an example of the XML output.

Figure 3.6: **XML Visualization.** This figure shows an example of XML Visualization. It is easy to copy this output and use it as input to applications that are able to parse XML data and use it for custom visualizations or data mining tools. This technique facilitates inter-application communication. The first entry in the XML output shown in this figure means that the average volume and occupancy for year 2002, in April, on Wednesdays, between 5am and 6am is 2 and 41 respectively.

3.4 Traffic Data Mining

Data mining is used to reveal novel patterns in the data. We used several data mining algorithms to find interesting patterns in the traffic data. These patterns can be used by traffic personnel to understand the traffic data and use the knowledge to make traffic-related decisions.

We implemented data mining algorithms from 3 different classes of data mining algorithms, classification, association analysis and clustering analysis.

3.4.1 Characterization

Characterization data mining algorithms are used to separate the data into meaningful classes. For example, the traffic data could be divided into four meaningful classes based on high and low values of volume and occupancy as described in Section 3.1.

An example task might be to find which class a sensor belongs given a group of sensors on a particular highway or on different sections of different highways. We could also characterize sensors according to their traffic conditions at different times of day, week or year.

We used the Self Organizing Maps [Koh82][Koh90] algorithm for characterization of sensors in the traffic data. The Self Organizing Map was proposed by Kohonen and has found a wide variety of application areas including speech recognition [KTS+87], sentence understanding [SJ90], optimization problems [AdlCVT88], etc.

The algorithm for the Self Organizing Maps is as follows:

- Initialize the weight vectors w_j for each output node to small random values. These are called *codebook vectors.*
- Initialize time t = 0.
- Repeat for a large number of iterations:
    1. Select an input sample i_l.
    2. Select best matching unit (BMU) as the output node having the minimum Euclidean distance from the current input sample.

$$BMU = min_j(\sum_{k=1}^{n}(i_{l,k}(t) - w_{j,k}(t))^2)$$

3. Update the weight for the BMU and its neighborhood using the update rule:

$$w_j(t+1) = w_j(t) + \eta(t)(i_l(t) - w_j(t))$$

where $\eta(t)$ is the learning rate at time t.

4. Increment t.

After training the network, the characterization is as simple as finding the nearest codebook vector to the input data point using the Euclidean distance.

3.4.2 Association Rule Extraction

Association rule mining searches for relationships between items in a dataset. We can find associations among sets of highways at different periods in time. For example, {Friday, 4-8pm, I-35, high} $\Rightarrow$ {Friday, 4-8pm, I-694, high} is an association rule which indicates that if traffic is high on I-35 on Friday evenings then it also high at the same time on I-694. Such rules can used to find out interesting and novel relationships between subsets of the data.

There has been a lot of work on association analysis since it was first introduced by Agrawal et al. [AIS93] along with the support-confidence framework and an algorithm to mine large itemsets called AIS after the initials of the authors. Mannila et al. [MTV94] further improved the AIS algorithm and introduced sampling from the database. Agrawal et al. [AS94] defined the popular Apriori algorithm that we have implemented in our work. Savasere et al. [SON95] developed the PARTITION algorithm that scanned the database only twice. Zaki [Zak00] introduced a number of new algorithms for association analysis. Recently, Han et al. [HPYM04] described a method called FP-growth that uses a FP-tree structure to store the database in a compressed form.

We introduced the two terms support and confidence in Section 2.3 that are used in association analysis. We introduce some more terms in this subsection and discuss the mathematical interpretation of support and confidence.

A set of items is referred to as an *itemset*. For example, the transactions that occur on a particular day in a company can be viewed as an itemset. In the case of the traffic data, the itemset could be the average volume and occupancy recorded for each sensor in a day. Each transaction in this set is an item. We note that any non-empty subset of this itemset is also an itemset. An itemset containing k items is called a *k-itemset*. If we call I the total set of items in the database D and A, B are itemsets such that $A \subset I$, $B \subset I$ then an association rule is defined as $A \Rightarrow B$. The association rule $A \Rightarrow B$ holds in D with support S and confidence C where:

$$S(A \Rightarrow B) = P(A \cap B)$$

$$C(A \Rightarrow B) = P(B|A)$$

$P(A \cap B)$ denotes the joint probability of occurrence of A and B in the database D and $P(B|A)$ denotes the conditional probability of occurrence of B given that A has occurred.

An itemset satisfying the minimum support count is called a *frequent itemset*. An association rule satisfying the minimum support count and the minimum confidence threshold is called a *strong association rule*.

Different types of association rules can be found. *Boolean association rules* find associations with respect to presence or absence of items. For example, buys(laptop) $\Rightarrow$ buys(optical-mouse) is a boolean association rule. Valiant [Val84] [Val85] has described discovering such rules in his work. *Quantitative association rules* find association rules between continuous valued attributes like age or income. For example, income(40K-60K) $\wedge$ age(25-30) $\Rightarrow$ owns(car). Srikant et al. [SA96] describe mining of quantitative association rules in their work.

Association rules can also be defined based on the number of dimensions involved in the rule. *Single dimensional association rules* involve only one dimension and *Multi-dimensional association rules* involve more than one dimension. For example, buys(milk) $\wedge$ buys(bread) $\Rightarrow$ buys(eggs) is a single dimensional association rule and doors(2) $\wedge$ cylin-

ders(8) $\Rightarrow$ price(high) is a multi-dimensional association rule. Kamber et al. [KHC97] discuss mining of multi-dimensional association rules using metarules. *Metarules* can be thought of as templates for discovering association rules that help in reducing the subset of data and improving the efficiency of mining. However, we implemented the Apriori algorithm for mining multi-dimensional association rules, instead of metarules.

The levels of abstractions in the association rules classify them into two general categories. *Single level association rules* contain attributes at a single level of abstraction. For example, day(Monday-Friday) $\wedge$ time(4) $\Rightarrow$ traffic(bad) has a single level of abstraction. *Multi-level association rules* can generalize the attributes and form rules at different levels of abstraction. For example, time(rush-hour) $\wedge$ hwy(I-35E) $\Rightarrow$ traffic(bad) aggregates the time to rush-hour and non-rush-hour and individual traffic sensors to highways. Algorithms for discovering multi-level association rules in databases have been described by Han et al. [HF95]. We limit our work to single level association rules.

The Apriori algorithm [AS94] is divided into two basic steps:

1. In the first step all the frequent k-itemsets are found.

2. In the second step, strong association rules are generated from the frequent k-itemsets.

The algorithm for the first step is as follows:

1. Generate the candidate itemsets C_1. All the 1-itemsets in the database form C_1. This step counts the number of occurrences of each item.

2. Determine the frequent itemsets L_1 as the candidate itemsets satisfying the minimum support count. In this step we eliminate all the 1-itemsets that do not satisfy the minimum support count.

3. Repeat the following steps k times until all the k-itemsets are below the minimum support threshold:

(a) Join two L_{k-1} sets to generate candidate k-itemset C_k. The two L_{k-1} sets are joinable if their first k-2 items are the same. A union is performed on these two sets to generate C_k.

(b) If C_k has any infrequent (k-1)-itemset then discard C_k. This is an optimization step to avoid unnecessary full table scans. If a set is infrequent then any superset of that set will also be infrequent. This property is known as the *Apriori property.* As a result of this property we do not need to count the number of occurrences for C_k if we know that some subset of C_k is infrequent.

(c) Determine the count (number of occurrences) for C_k if no subset of C_k is infrequent.

(d) If count for C_k is greater than minimum support count then generate L_k.

4. Return L_k for determining the strong association rules. This is used in the second step of the algorithm to generate the association rules.

The algorithm for the second step is as follows:

1. Generate all possible non-empty subsets for each frequent itemset L_k.

2. Generate a rule A $\Rightarrow$ B for every subset A of L_k where B = L_k - A, if $\frac{support\_count(L_k)}{support\_count(A)} \geq minimum\_confidence$.

In the second step of the algorithm we first generate all the non-empty subsets for each of the k-frequent itemset generated by step 1. This can be done by finding all combinations of items in the k-frequent itemset starting from subsets with size 1 and continuing till subsets having size k-1. For each subset A and its complement B we generate a rule A $\Rightarrow$ B if it satisfies the minimum confidence threshold as shown in Step 2.2 of the algorithm.

To implement the association analysis we query the database along with its attributes to discover multi-dimensional association rules. For example, if we would like to discover

the relationship between time of the day and the traffic conditions for different sensors then we issue the following query:

```
SELECT SR_ID, TOD_ID, AVG(VOLUME), AVG(OCCUPANCY)
FROM TRAFFIC_DATA
GROUP BY SR_ID, TOD_ID;
```

This query will return the average volume and occupancy for all times of day for each sensor. We then use this data and apply the Apriori algorithm as described above to discover the frequent itemsets and then form strong association rules from them. Then we sort the association rules according to their confidence level and present it to the user.

Our tool currently does not discover multi-level association rules automatically from the dataset. For example, if we wish to discover the association rules for traffic conditions on different highway for weekdays and weekends we issue the following query:

```
SELECT SR_HIGHWAY, DOW_WEEKDAY_FLAG, VOLUME, OCCUPANCY
FROM TRAFFIC_DATA, SENSOR, DAY_OF_WEEK
WHERE TRAFFIC_DATA.SR_ID=SR_ID AND TRAFFIC_DATA.DOW_ID=DAY_OF_WEEK.DOW_ID
GROUP BY SR_HIGHWAY, DOW_WEEKDAY_FLAG;
```

This query joins the fact table TRAFFIC_DATA with the DAY_OF_WEEK table and SENSOR table and groups by SR_HIGHWAY and DOW_WEEKDAY_FLAG to give the average volume and occupancy for each highway for weekdays and weekends. Using this data as input to the Apriori algorithm we are able to discover multi-level rules.

3.4.3 Clustering

Clustering is an unsupervised method for classifying data. When we have training data available (i.e., data points with labels for the class that they belong to), then we can use supervised approach to classify the data. However in absence of such data we use unsupervised methods for classification. Unsupervised methods should also be used if we do not know the number of classes in the data. It is important to note that though clustering

analysis produces clusters of the data that are related, it does not define the label of the clusters. Hence we are left with the task of labeling the data points in a cluster.

Clustering methods can be divided into two broad categories: agglomerative and divisive. Agglomerative clustering methods find the most similar pair of data points and group them together to form a new data point and repeat this process until it ends one single data point which is the top most cluster. We can think of agglomerative clustering methods as a bottom-up approach. Divisive clustering methods start by dividing the whole data into a certain number of groups and then reform the groups to classify the most similar set of data points into the same cluster. Divisive clustering can be thought of as a top-down approach. In our work we implement the K-means clustering algorithm that is an example of divisive clustering algorithm.

The K-means algorithm uses K as the number of clusters desired. K-means [JMF99] could work as follows:

1. Initialize K centroids.

2. Determine the closest centroid to each data point and assign the data point to the closest centroid.

3. For each centroid, calculate a gradient consisting of the sum of the differences between that centroid and the points that make up its cluster.

4. Move the centroids towards the center of the cluster by a taking a step in the gradient direction based on the gradient calculated in Step 3 and a movement or learning rate.

5. Repeat Steps 2 through 4 for a fixed number of epochs or until all the centroids do not show a significant change from the previous values.

The centroid in the K-means method is generally the mean position of all the points of the cluster, hence the name K-means. The distance or similarity between the centroid and

the data point can be determined using various measures and metrics. The distance measure determines the distance between the data points in the feature space and the similarity measures or metrics determine the similarity between the two data points. Measures give the exact quantitative distance or similarity between the data points and provide an indication as to what degree the two data points are similar or far apart.

There are several distance measures each having their own limitations and advantages. Some of the examples of distance measures are Manhattan distance, Euclidean distance, Minkowski distance, Pearson's correlation and cosine correlation. We used the Euclidean distance in our implementation.

The Euclidean distance specifies the distance between two data points. The Euclidean distance between two points i and j can be found using the following formula:

$$D_{euc}(i,j) = \sqrt{\sum_{f=1}^{|features|} (x_{i,f} - x_{j,f})^2}$$

If a feature has very large values like a person's salary as compared to features that have smaller values like a person's age then it will dominate the distance and will produce false results. Hence, it is necessary to normalize the data before we cluster it using K-means. A data point $x_{i,j}$ can be normalized using the following formula:

$$Norm(x_{i,f}) = \frac{x_{i,j} - Min(x_{*,j})}{Max(x_{*,j})}$$

where, $Min(x_{*,j})$ means the minimum value in feature j and $Max(x_{*,j})$ means the maximum value in feature j. The missing values in the data may be replaced by the average value for the corresponding feature.

Chapter 4

Experiments

This chapter will discuss the experimental results for each part of our implementation. First, we discuss the results of the data warehouse with respect to optimization and discovering interesting patterns by directly querying the data warehouse. We then describe how novel patterns can be found using the visualization techniques and present some of the results that we found. Finally, we describe how the different data mining algorithms can be used to discover interesting patterns in the traffic data and present some of the results.

4.1 Data Warehouse Implementation Details

4.1.1 Optimization

Our primary optimization technique, as discussed in Subsection 3.2.5, is the use of indexes on foreign keys in the TRAFIC_DATA table namely, TOD_ID, DOW_ID, TOY_ID, YR_ID, SR_ID and LOC_ID. Each of these are primary keys in the dimension tables as shown in Figure 3.1.

We performed experiments to see the results of optimization by recording the query timings before and after the indexes were created. We used queries spanning different number of dimensions to justify the creation of indexes. For example,

```
SELECT TOD_ID, AVG(VOLUME), AVG(OCCUPANCY)
FROM TRAFFIC_DATA
GROUP BY TOD_ID;
```

is an example of a query spanning a single dimension.

```
SELECT SR_ID, LOC_ID, YR_ID, TOY_ID, DOW_ID, TOD_ID,
       AVG(VOLUME), AVG(OCCUPANCY)
FROM TRAFFIC_DATA
GROUP BY SR_ID, LOC_ID YR_ID, TOY_ID, DOW_ID, TOD_ID;
```

is an example of a query spanning all the six dimensions and

```
SELECT AVG(VOLUME), AVG(OCCUPANCY)
FROM TRAFFIC_DATA;
```

is an example of a query spanning zero dimensions. We note that the query spanning all six dimensions is not necessary for our analysis, because it simply returns all the rows in the data warehouse without any summarization. Hence the use of indexes does not result in any particular improvement for querying the whole dataset however it may offer improvements if we are looking for a subset of the data involving all six dimensions.

The timings of queries spanning various number of dimensions before and after index creation are presented in Table 4.1. The results clearly indicate that the optimization was very useful in reducing the query timings drastically. We observe that the increase in query timing is linear with respect to increase in the number of dimensions after indexing. The increase in query timing before indexing is exponential with respect to increase in the number of dimensions. This is shown in Figure 4.1.

We note that the query timing for zero dimensions is similar because both the queries have to do a full table scan and indexes do not offer much help.

Table 4.1: **Results of Data Warehouse Optimization** This table shows average timings of queries (in seconds) spanning different number of dimensions before and after indexing. It also shows the percentage by which the query timings reduced.

Number of Dimensions	Time before indexing	Time after indexing	% reduction
0	53	43	18.87
1	464	153	67.03
2	745	193	74.09
3	1438	222	84.56
4	2972	265	91.08
5	5840	317	94.57

4.1.2 Finding Patterns by Querying

It is possible to find simple patterns in the traffic data by directly querying the data warehouse. It is difficult to find more complex patterns using this method because the results are presented in a tabular form and finding the patterns in a large amount of data is difficult to the unaided human eye.

We present some queries that find some simple patterns in the traffic data. An example of such a query and its results are as follows:

```
mysql> SELECT DOW_ID, AVG(VOLUME), AVG(OCCUPANCY)
    -> FROM TRAFFIC_DATA GROUP BY DOW_ID;

+--------+-------------+----------------+
| DOW_ID | AVG(VOLUME) | AVG(OCCUPANCY) |
+--------+-------------+----------------+
|      1 |      2.5866 |        67.8902 |
|      2 |      3.8004 |        94.3914 |
|      3 |      4.0206 |        99.0796 |
|      4 |      4.0823 |       101.8176 |
|      5 |      4.1057 |       101.8224 |
|      6 |      4.2096 |       101.6850 |
|      7 |      3.1149 |        76.5707 |
+--------+-------------+----------------+
```

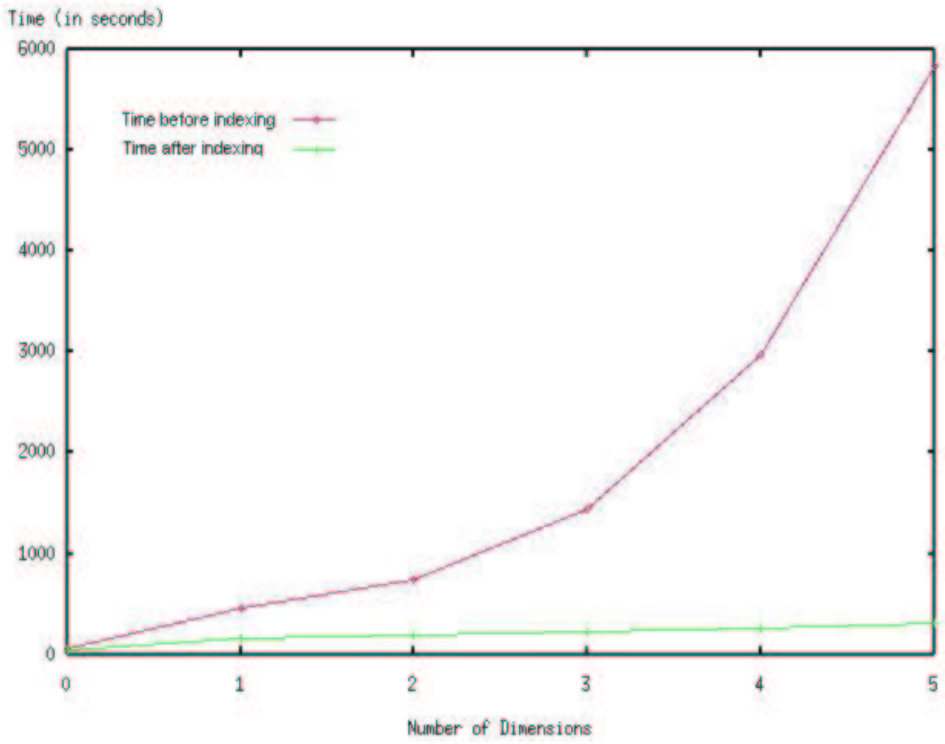

Figure 4.1: **Query timing before and after indexing.** This figure shows how the query timing varies with the increase in number of dimensions. It shows the query timing before and after indexing. The increase in query timing before indexing is approximately exponential and that after indexing is approximately linear.

The above query finds the average volume and occupancy for each day of the week. The results of this query indicate that the traffic on weekends is considerably less than weekdays. We also note that the volume is highest on Fridays which may be due to many people leaving for weekends around the same time. Interestingly, the volume and occupancy on Mondays are lower than the other weekdays which may indicate that people are still getting into gear for work and travel at a slightly different time than their usual work time.

To find out how the average traffic varies across each hour of the day we can issue the following query:

```
mysql> SELECT TOD_ID, AVG(VOLUME), AVG(OCCUPANCY)
    -> FROM TRAFFIC_DATA
    -> GROUP BY TOD_ID;
```

```
+--------+-------------+----------------+
| TOD_ID | AVG(VOLUME) | AVG(OCCUPANCY) |
+--------+-------------+----------------+
|      1 |      0.9609 |        45.8675 |
|      2 |      0.6521 |        41.9802 |
|      3 |      0.3425 |        38.0259 |
|      4 |      0.2672 |        37.2560 |
|      5 |      0.4874 |        40.1429 |
|      6 |      1.7762 |        57.7023 |
|      7 |      4.1819 |        96.4724 |
|      8 |      5.4065 |       128.7816 |
|      9 |      5.0820 |       121.6173 |
|     10 |      4.4634 |       103.3443 |
|     11 |      4.4399 |       100.8974 |
|     12 |      4.9462 |       108.6018 |
|     13 |      5.2857 |       113.8275 |
|     14 |      5.3491 |       115.1196 |
|     15 |      5.7878 |       122.6101 |
|     16 |      6.4117 |       139.2844 |
|     17 |      6.6414 |       151.0700 |
|     18 |      6.4280 |       151.6199 |
|     19 |      5.3610 |       116.7588 |
|     20 |      3.9845 |        90.5778 |
|     21 |      3.3511 |        80.9598 |
|     22 |      3.1821 |        78.7726 |
|     23 |      2.4562 |        67.9376 |
|     24 |      1.6246 |        56.2240 |
+--------+-------------+----------------+
```

We can determine the rush-hours using the results of this query. The rush-hours are between 8 and 10 in the morning and between 3 and 7 in the evening. We note the gradual decrease in traffic from 8pm till midnight and the sparse traffic after midnight until 7am.

The following query will reveal which highway has the worst traffic:

```
mysql> SELECT SR_HIGHWAY, AVG(VOLUME), AVG(OCCUPANCY)
    -> FROM TRAFFIC_DATA, SENSOR
    -> WHERE TRAFFIC_DATA.SR_ID = SENSOR.SR_ID
    -> GROUP BY SR_HIGHWAY;
```

SR_HIGHWAY	AVG(VOLUME)	AVG(OCCUPANCY)
I-35E	5.3987	107.3904
I-35W	3.1816	106.8124
I-394	5.1808	139.3663
I-494	5.2280	119.0704
I-694	2.7298	95.6496
I-94	5.0245	117.9536
MN-100	7.0285	147.4180
MN-36	4.1023	98.4920
MN-62	6.6568	146.2213
NULL	3.4653	87.9540
TH-212	8.2671	138.9037
US-169	5.0940	94.0925

From the results we can see that trunk highway 212 (TH-212) has the highest volume and MN-100 has the highest occupancy. We note that the sensors for which we do not know the highway they are placed on are classified in the category NULL.

To determine which highway has the worst traffic we define a measure called occupancy-volume (O/V) ratio as:

$$O/V ratio = \frac{Occupancy}{Volume}$$

This ratio indicates how much fraction of the time each car spent on the sensor during the 30 seconds. The highway having the highest O/V ratio can be declared as the one having the worst traffic conditions. The O/V ratio for each highway is shown in Table 4.2. From the table it is easy to determine that I-694 has the worst traffic conditions and trunk highway 212 (TH-212) has the best traffic conditions.

To summarize, querying the data warehouse directly is a simple yet powerful tool to answer questions regarding traffic conditions for users who are familiar with the structured query language (SQL) and have direct access to the data warehouse. However, it may be difficult and time-consuming to discover interesting patterns from large amounts of tabular

Table 4.2: **Occupancy-Volume Ratio for Highways** This table shows the average volume, occupancy and O/V ratio for the highways. We have categorized the sensors for which we do not know the highway that they are placed on into NULL. The results indicate that I-694 has the worst traffic conditions and trunk highway 212 (TH-212) has the best traffic conditions.

Highway	Average Volume	Average Occupancy	O/V Ratio
I-35E	5.3987	107.3904	19.8919
I-35W	3.1816	106.8124	33.5719
I-394	5.1808	139.3663	26.9005
I-494	5.2280	119.0704	22.7755
I-694	2.7298	95.6496	35.0391
I-94	5.0245	117.9536	23.4757
MN-100	7.0285	147.4180	20.9743
MN-36	4.1023	98.4920	24.0089
MN-62	6.7806	150.1936	22.1505
NULL	3.4771	87.1829	25.0735
TH-212	8.2671	138.9037	16.8020
US-169	5.0940	94.0925	18.4712

data with the unaided human eye. So direct querying may not be very useful to discover complex relationships.

4.2 Data Visualization

The data visualization techniques can be used individually as well as they can be used together to find useful patterns. We performed experiments to use these visualization techniques to demonstrate how interesting patterns can be found in the traffic data. We describe some of the results in this section.

Map visualization is useful to determine a starting point for analysis. Given a subset of the data it allows the user to get a glimpse of the distribution of the data temporally as well as capturing the spatial aspect of the data. A user can view the traffic distribution on the highway of his interest by using this technique. For example, we were interested in the traffic distribution of TH-212 and wanted to know how the traffic conditions were at 8am in different months in the year 2002. Figure 4.2 shows the traffic conditions on highway TH-212 from March 2002 to June 2002 on Mondays at 8am. The results indicate that a couple of sensors show low volume and occupancy in March and June on Mondays at 8am. We can then use other visualizations to determine how the traffic conditions on those two sensors differ with respect to the other sensors on TH-212 and to discover how the traffic conditions on those two sensors vary at 8am with respect to different months for all days of the week. Thus, map visualization gave us a good starting point to find out more about the traffic distribution of sensors on highway TH-212.

We wanted to determine the traffic conditions of all the sensors on TH-212 with respect to each other for different months in 2002 for Mondays at 8am. We used the bar graph visualization to perform this comparison. Figure 4.3 shows the results on this analysis. The results show that both volume and occupancy on sensor 305 are low in March 2002 on Mondays at 8am and on sensor 309 in June 2002 on Mondays at 8am.

We further investigate the traffic conditions on sensors 305 and 309 for different months in 2002 for all days of the week. Figure 4.4 shows the traffic conditions on sensors 305 and 309 at 8am for all days on the week from January 2002 to June 2002. The results show that the traffic is lower than usual for sensor 305 in March 2002 for all days of the week and it is very low on sensor 309 in the June 2002 for all days of the week. We would then like to find out whether the traffic is low only at 8am or for all times of the day. We show the results of this analysis using heat map visualization in Figure 4.5. We limit the results to Wednesday and Thursday in March and June 2002 for the sake on clarity in the diagram. The results show that volume was low in March 2002 for all times of the day on sensor 305.

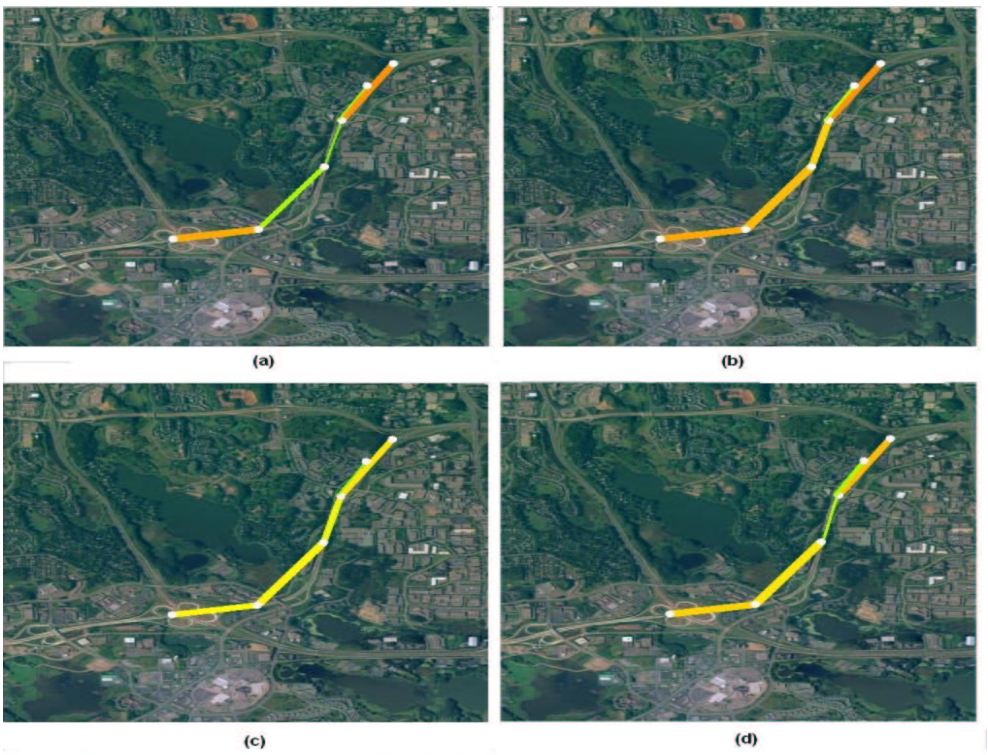

Figure 4.2: **Traffic condition on TH-212 from March 2002 to June 2002 on Mondays at 8am.** This figure shows the traffic condition on TH-212 from March 2002 to June 2002 on Mondays at 8am. (a) March (b) April (c) May (d) June. The results indicate low traffic condition on a couple of sensors on TH-212 for March and June 2002 on Mondays at 8am.

The results also indicate that the volume was low in June 2002 for all times of the day on sensor 309. We can use this analysis to investigate whether the low traffic conditions on these sensors were due to sensor malfunction or due to some other reason. Thus we can use visualizations to identify malfunctioning sensors and anomalies in traffic conditions.

Line graph visualization helps the user to determine patterns as well as find outliers in the data. It is easier to find subtle dissimilarities in patterns of the data using line graph visualization than the heat map or the bar graph visualization. We use the line graph visualization to find patterns in the average traffic data per hour for three highways. Figure 4.6(a) shows the distribution of the average traffic per hour for TH-212, I-35 and I-94. We see that the traffic shows a similar distribution with respect to hour of the day

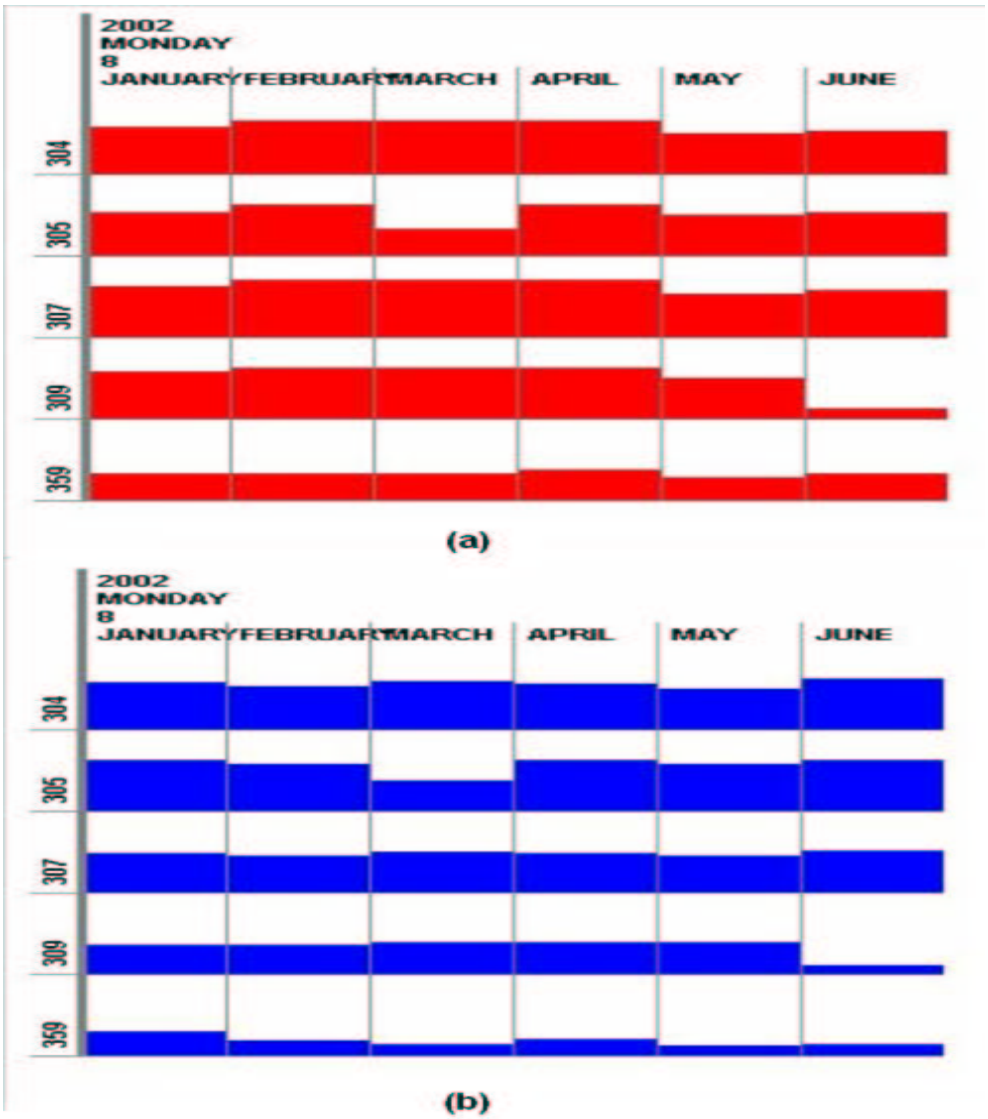

Figure 4.3: **Traffic conditions of all sensors on TH-212 from January 2002 to June 2002 on Mondays at 8am** This figure shows the (a) Volume (b) Occupancy for all sensors on TH-212 from January 2002 to June 2002 on Mondays at 8am. The results show that sensor 305 has low traffic in March 2002 and sensor 209 has low traffic in June 2002 on Mondays at 8am.

regardless of the highway. We conclude that there exists a strong correlation between hour of the day and the volume and occupancy for a highway from the results. We further want to explore whether there is a correlation between day of the week and the traffic data.

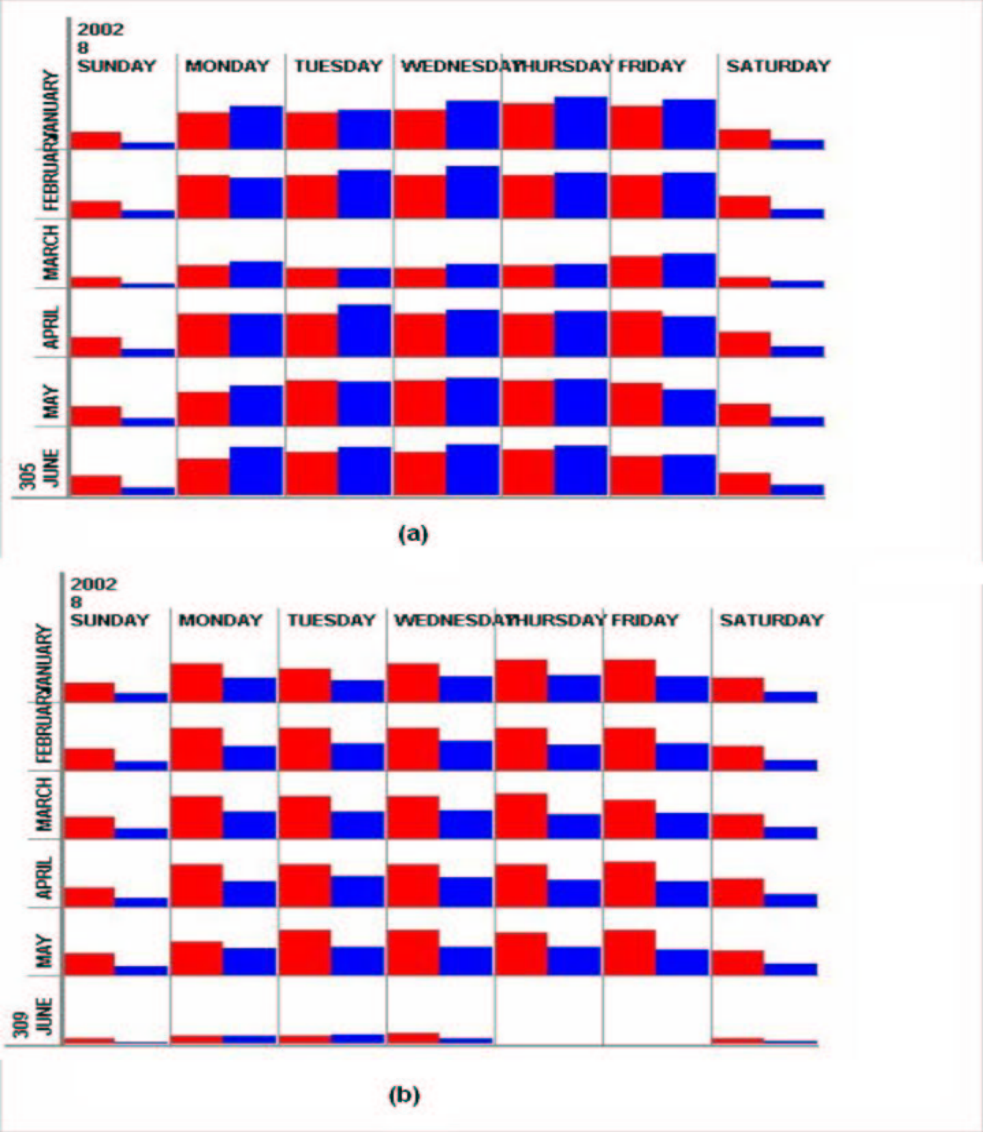

Figure 4.4: **Traffic conditions on sensor 305 and 309 for all days of the week at 8am** This figure shows the traffic conditions on (a) sensor 305 (b) sensor 309 for all days of the week during January 2002 and June 2002 at 8am. The figure indicates that the traffic was low at 8am for all days of the week on sensor 305 during March and was low on all days of the week on sensor 309 during June 2002.

Figures 4.6(b), 4.6(c) and 4.6(d) show the hourly traffic distribution for each day of the week respectively for highways TH-212, I-35 and I-94. The results show that there exists a strong correlation between day of the week and the traffic distribution. In particular, all

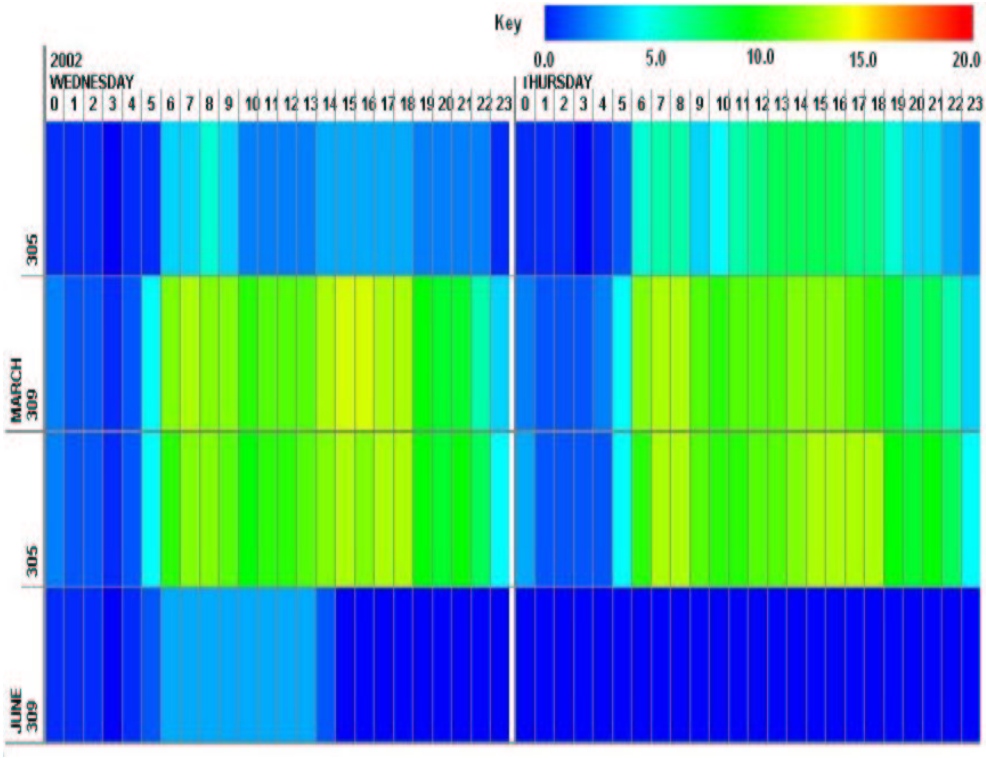

Figure 4.5: **Volume on sensors 305 and 309 for all times of the day on Wednesday and Thursday of March 2002 and June 2002** This figure shows the volume on sensors 305 and 309 for all times of the day on Wednesday and Thursday of March and June 2002. It shows that the volume was low on sensor 305 for all times of the day in March 2002 and volume was low on sensor 309 for all times of the day in June 2002.

weekdays show similar traffic distribution and weekends show similar traffic distribution.

We also note that in all cases the volume and occupancy show a similar distribution so we can also conclude that volume and occupancy are strongly correlated.

We were curious to know whether a similar pattern exists for individual sensors on the highways. Figure 4.7(a) shows the hourly occupancy of some of the individual sensors on TH-212 for Sunday and Monday. Figure 4.7(b) shows the hourly occupancy of some of the individual sensors on I-35 for Sunday and Monday. From the results we conclude that the sensors show a similar pattern, but there exist subtle changes in the distributions for individual sensors on the highways.

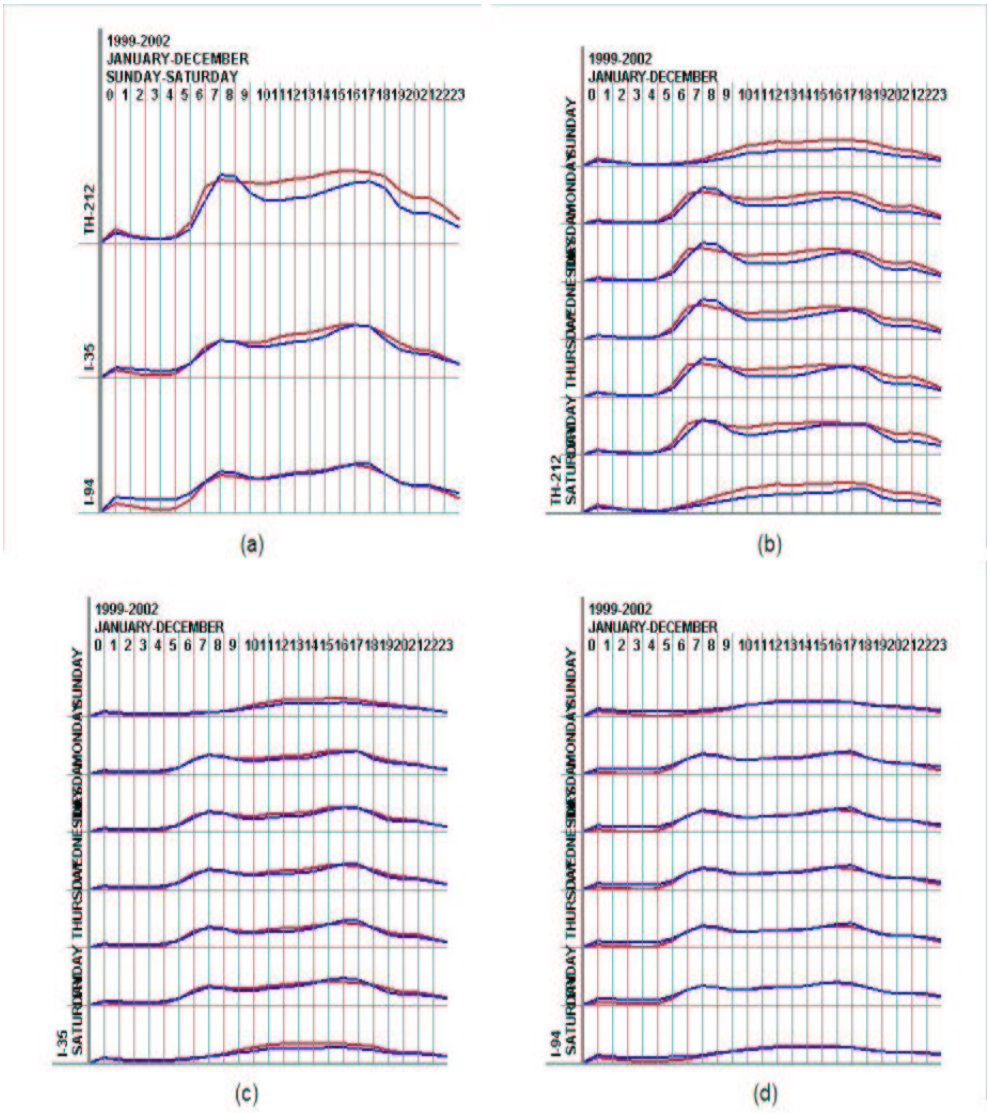

Figure 4.6: **Hourly traffic patterns using line graph visualization.** (a) This figure shows the average hourly traffic for each highway using line graph visualization. (b) This figure shows the hourly traffic for each day of the week for highway TH-212. (c) This figure shows the hourly traffic for each day of the week for highway I-35. (d) This figure shows the hourly traffic for each day of the week for highway I-94.

Maclin et al. [MCC03] were able to achieve an average of 77% compression of the traffic data using lossy algorithms due to the strong correlations in the data and hour of the day.

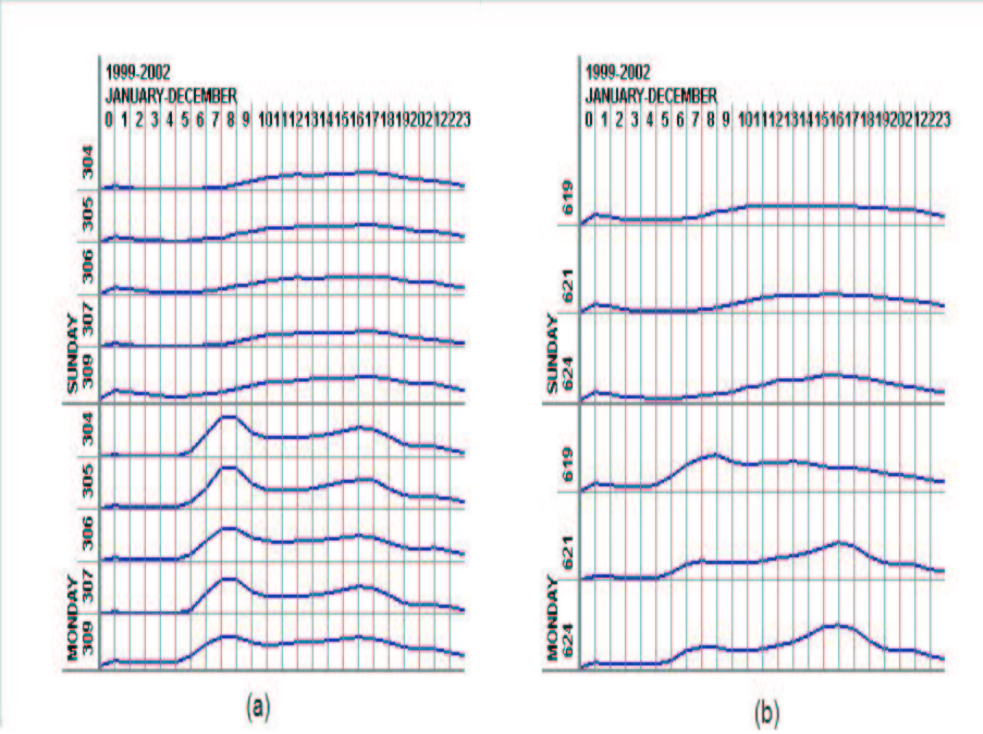

Figure 4.7: **Average occupancy of individual sensors per hour for Sunday and Monday.** (a) Average occupancy of individual sensors per hour on highway TH-212 for Sunday and Monday. (b) Average occupancy of individual sensors per hour on highway I-35 for Sunday and Monday.

4.3 Data Mining

4.3.1 Characterization

We performed experiments with the Self Organizing Maps data mining tool to characterize the data with respect to the dimensions in the database. We characterized the data into four classes based on the high and low values of volume and occupancy. Class 1 denotes low volume and low occupancy and signifies absence of traffic. Class 2 denotes low volume and high occupancy and signifies bad traffic condition. Sensors having high volume and low occupancy are characterized as class 3 and those having both high volume and occupancy are characterized as class 4. Class 3 denotes smooth traffic conditions on the sensors and

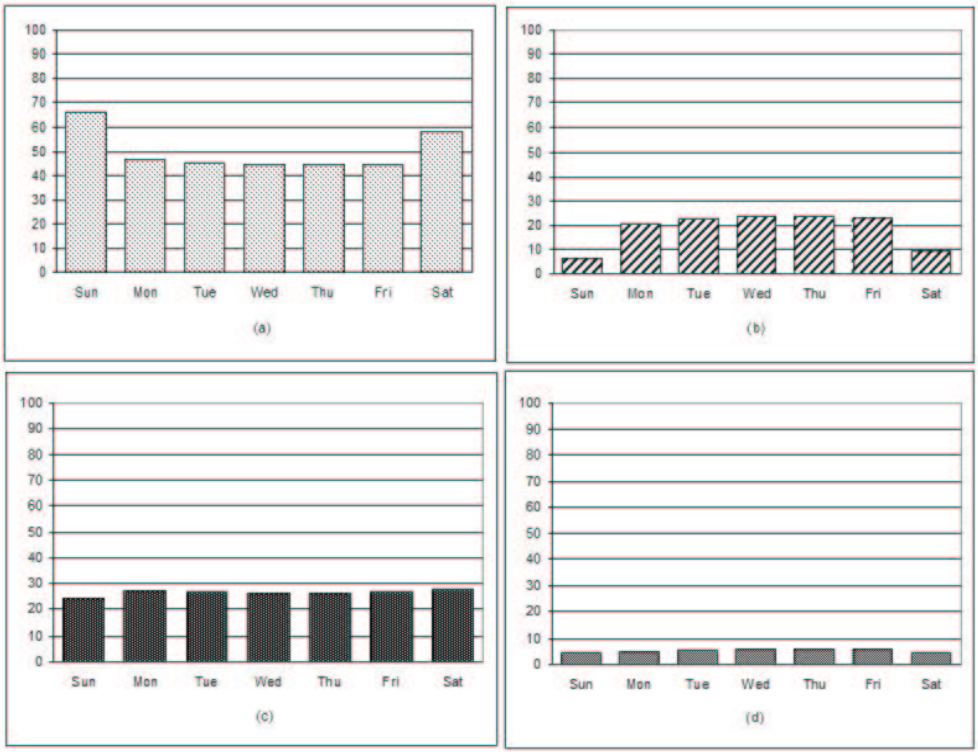

Figure 4.8: **Sensor class distribution for each day of week.** This figure shows the distribution of the sensor classes with respect to each day of the week. (a) Class 1 distribution. (b) Class 2 distribution. (c) Class 3 distribution. (d) Class 4 distribution.

class 4 denotes heavy traffic conditions on the sensors.

Figure 4.8 shows the distribution of classes for each day of the week. Figures 4.8(c) and (d) show that the distribution of classes 3 and 4 is almost similar for all days of the week. Figure 4.8(b) shows that there are lower number of class 2 sensors on weekends and there is a proportionate increase in the number of sensors for class 1 on weekends as shown in Figure 4.8(a). This means that the sensors having bad traffic on weekdays have an absence of traffic on weekends.

The hourly distribution of the classes is shown in Figure 4.9. Figure 4.9(a) shows that there is a significant rise in the number of sensors belonging to class 1 after 8pm and it stays high until 8am. This indicates a low traffic period for a lot of sensors. Approximately 20 to 30 percent of these sensors then become class 2 sensors and roughly an equal number

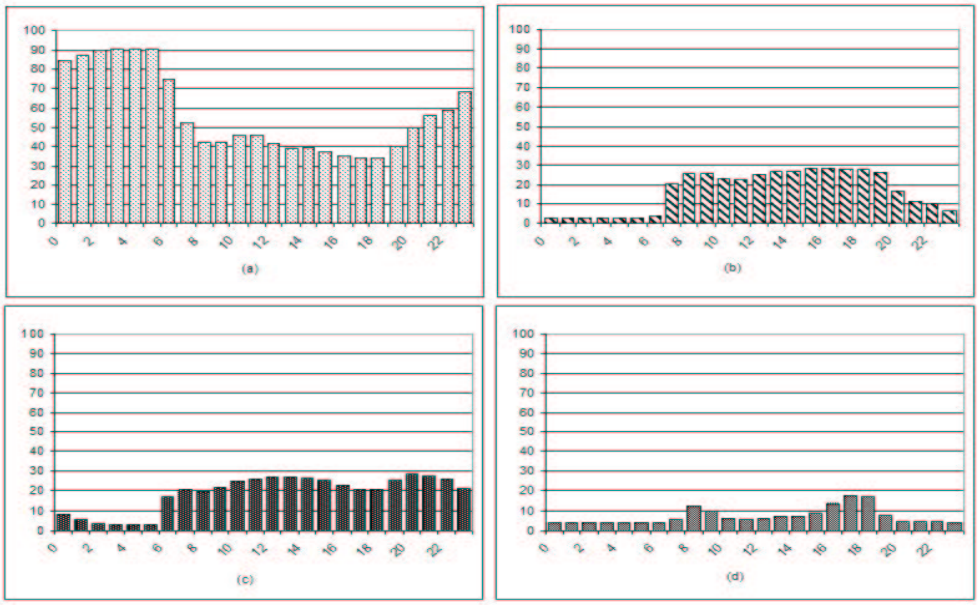

Figure 4.9: **Sensor class distribution for each hour of the day.** This figure shows the distribution of sensors classes with respect to each hour of the day. (a) Class 1 distribution. (b) Class 2 distribution. (c) Class 3 distribution. (d) Class 4 distribution.

of sensors become class 3 sensors between 8am and 8pm as shown in Figure 4.9(b) and (c). This indicates that 20 to 30 percent of the sensors have bad traffic conditions on them and an equal number of sensors have good traffic conditions on them. About 10 to 15 percent of sensors experience heavy traffic during rush hours as indicated by Figure 4.9(d).

4.3.2 Association Analysis

We used the Apriori algorithm [AIS93] to discover association rules from the data. We experimented with various subsets of the data and using different combinations of minimum support and confidence to discover interesting rules in the data. The interesting findings of the research are described below. The algorithm we implemented discovers multi-dimensional as well as multi-level association rules as described in Section 3.4.2. The only limitation we impose on our algorithm is its ability to deal with continuous values. Instead of binning the volume and occupancy in the algorithm we binned them manually

Table 4.3: **Limits used for binning volume and occupancy.** This table shows the limits used for binning volume and occupancy into three bins called low, moderate and high based on their values.

	Bin	Lower Limit	Upper Limit
Volume	Low	0	5
	Moderate	5	15
	High	15	40
Occupancy	Low	0	25
	Moderate	25	150
	High	150	1800

into three bins each namely: low, moderate and high. The limits for the bins for volume and occupancy are shown in Table 4.3. These limits were designed manually based on some observations of the data.

One of the most interesting finding is the strong correlation between volume and occupancy. This means that as number of cars go up the fraction of time occupied on the sensors increases. The rules that support this claim are summarized in Table 4.4.

We also found that sensor number 305 that is placed on highway TH-212 has moderate volume with a confidence of 0.7 and a high occupancy with confidence 0.57. The rules are as follows:

$$Sensor = 305 \Rightarrow Volume = moderate$$

$$Sensor = 305 \Rightarrow Occupancy = high$$

Some of the general rules that we found in our experimentation are as follows:

Day of week = 1 $\Rightarrow$ Volume = low (Confidence = 0.65)

Day of week = 2 $\Rightarrow$ Occupancy = moderate (Confidence = 0.57)

Day of week = 3 $\Rightarrow$ Occupancy = moderate (Confidence = 0.56)

Table 4.4: **Association Rules for Volume-Occupancy Correlation.** This table shows the rules that indicate that there is a strong correlation between volume and occupancy. The minimum support used to discover these rules is 0.05 and confidence threshold is 0.5. The first column describes the data subset used to discover the rules.

Data Subset	Association Rule	Confidence
TH-212, Year 2002	Volume = high ⇒ Occupancy = high	1.00
	Occupancy = low ⇒ Volume = low	1.00
	Occupancy = high ⇒ Volume = moderate	0.67
I-35, Year 2002	Occupancy = low ⇒ Volume = low	1.00
	Occupancy = high ⇒ Volume = moderate	0.92
	Volume = moderate ⇒ Occupancy = moderate	0.58
All highways, Year 2002	Occupancy = low ⇒ Volume = low	1.00
	Occupancy = high ⇒ Volume = moderate	0.99

Day of week = 4 ⇒ Occupancy = moderate (Confidence = 0.55)

Day of week = 5 ⇒ Occupancy = moderate (Confidence = 0.54)

Day of week = 6 ⇒ Occupancy = moderate (Confidence = 0.54)

Day of week = 7 ⇒ Volume = low (confidence = 0.57)

The above rules simply state that the volume is low on weekends and occupancy is high on weekdays.

Some of the interesting rules that we discovered for different highways are shown in Table 4.5.

We can use these rules to determine which factors in the data are related to each other and with what confidence. This analysis can help in discovering interesting and unknown relationships in the data.

Table 4.5: **Association Rules for Highways.** This table shows the interesting rules related to highways that we discovered and their confidence.

Association Rule	Confidence
Highway = I-694 ⇒ Volume = low	0.98
Highway = I-694 ⇒ Occupancy = moderate	0.69
Highway = I-694, Occupancy = moderate ⇒ Volume = low	0.98
Highway = I-35 ⇒ Occupancy = moderate	0.77
Highway = I-35 ⇒ Volume = low	0.73
Highway = MN-36 ⇒ Occupancy = moderate	0.74
Highway = MN-100 ⇒ Volume = moderate	0.69
Highway = MN-62 ⇒ Volume = moderate	0.68

4.3.3 Clustering

We performed clustering on different subsets of data to determine their distribution and the clusters that they form. We were mainly interested in how the sensors cluster together. Figure 4.10 shows the clustering results for the average volume and occupancy for all sensors. The results show that only 3 clusters were formed. Most of the sensors have low occupancy and are clustered in cluster 2. There are 6 sensors with very high occupancy and moderate volume and are clustered in cluster 1. The third cluster consists of sensors having low volume and moderate occupancy. There are 103 sensors in this cluster and 87 of these sensors do not have a highway associated with them.

We also wanted to find out how the distribution of the sensors in clusters change with respect to rush-hours and non rush-hours. Figure 4.11 shows the distribution of sensors in clusters for rush-hours and non rush-hours. Cluster 1 represents sensors with low volume, low occupancy. Cluster 2 represents sensors with moderate volume, high occupancy. Cluster 3 represents sensors with moderate volume and low occupancy. Cluster 4 represents sensors

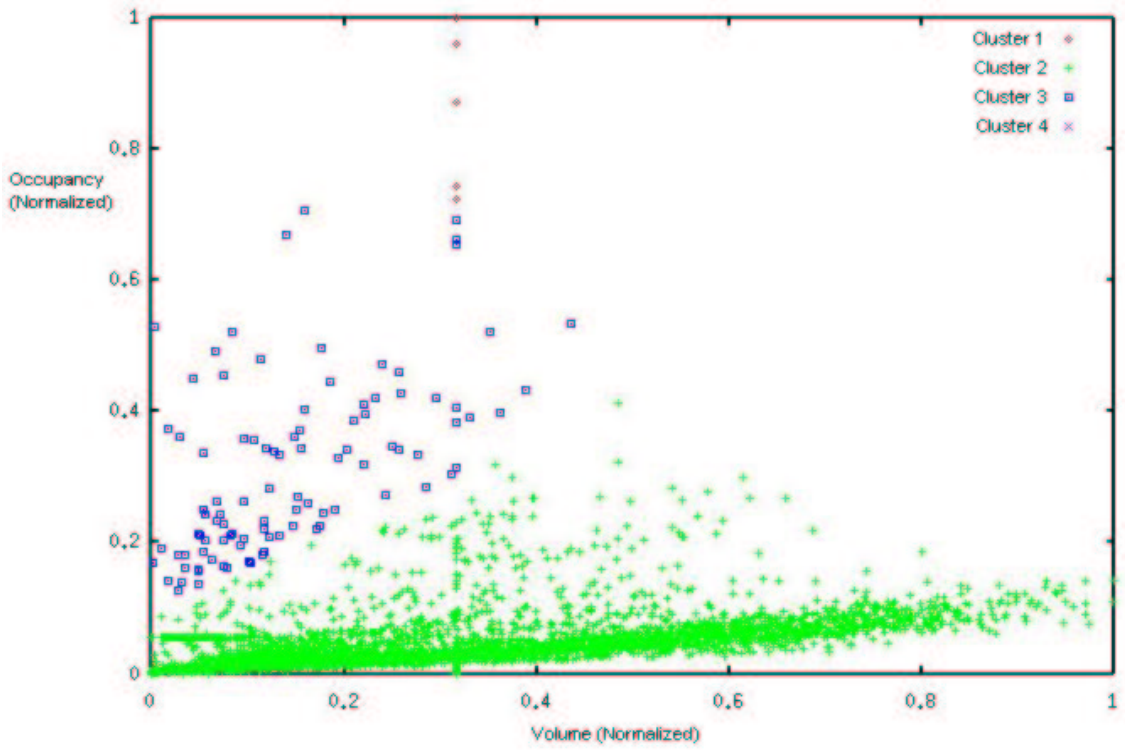

Figure 4.10: **Clusters of Sensors.**

with high volume and low occupancy. The figure shows a rise in the number of sensors in cluster 4 from non rush-hours to rush-hours. This means that there are more sensors having high volume and low occupancy in rush-hours than in non rush-hours.

We also analyzed how the number of sensors in each cluster changes for each highway for rush-hours and non rush-hours. All the highways show a rise in the number of sensors belonging to cluster 4 which means that the traffic increases on all highways during rush-hours. In particular, MN-62 shows a significant increase in sensors belonging to cluster 4. The number of sensors belong to cluster 4 during non rush-hours are 18 out of a total of 41 sensors on MN-62. During rush-hours the number of sensors belonging to cluster 4 increase to 30. We can conclude that MN-62 shows a significant change in traffic conditions from non rush-hours to rush-hours and take appropriate actions to improve the traffic conditions.

Figure 4.12 compares the clusters of sensors during weekends to those during weekdays.

There is a significant change in the number of sensors in cluster 3. Cluster 3 represents sensors having very high volume and moderate occupancy. There are 109 sensors in this cluster on weekdays while there is no sensor in this cluster on weekends. This means that the volume is lower on weekends.

In summary, these experiments demonstrate that it is simple using this tool to perform complex and interesting data analyses.

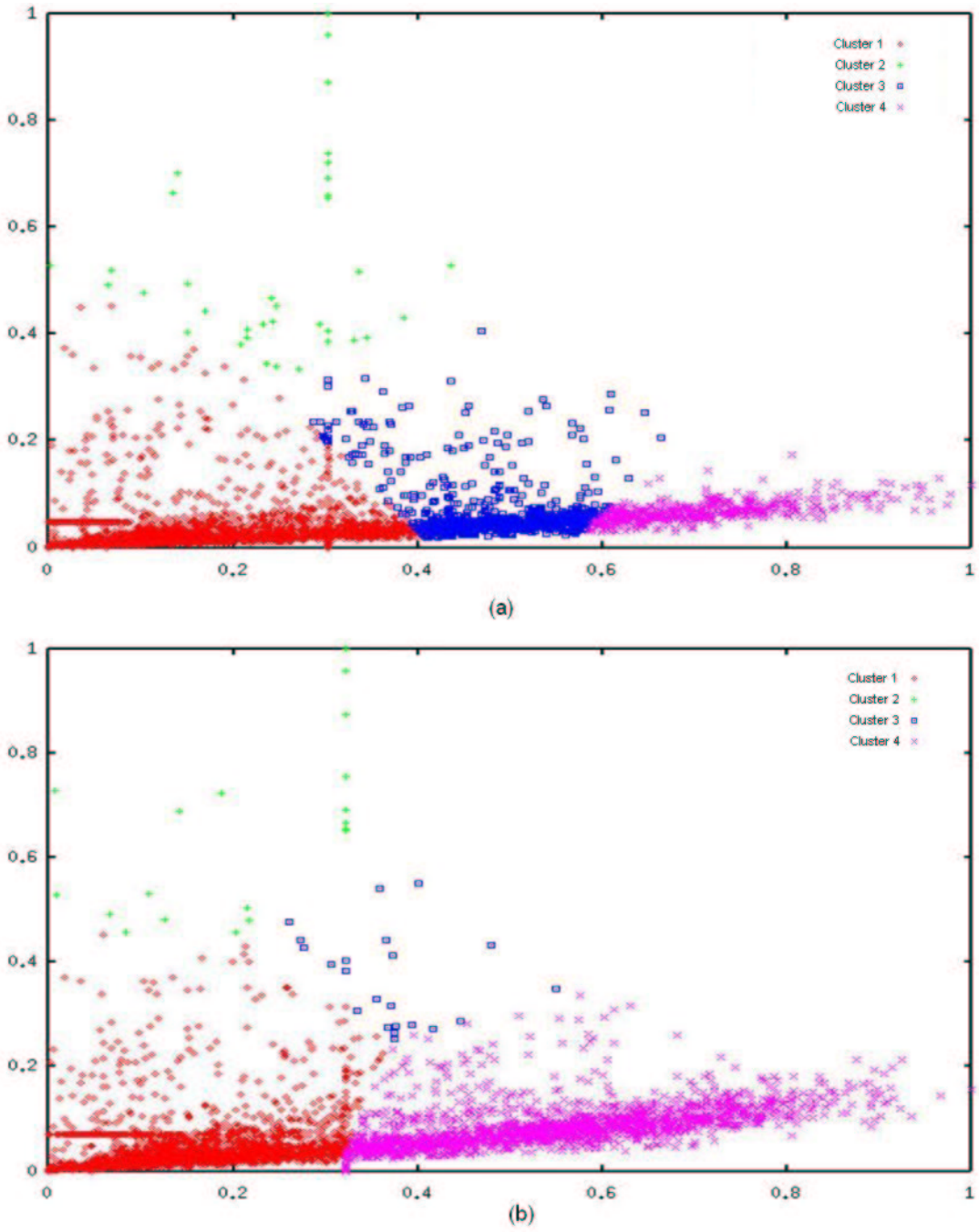

Figure 4.11: **Clusters of Sensors for rush-hours and non rush-hours.** (a) Non rush-hours. (b) Rush-hours.

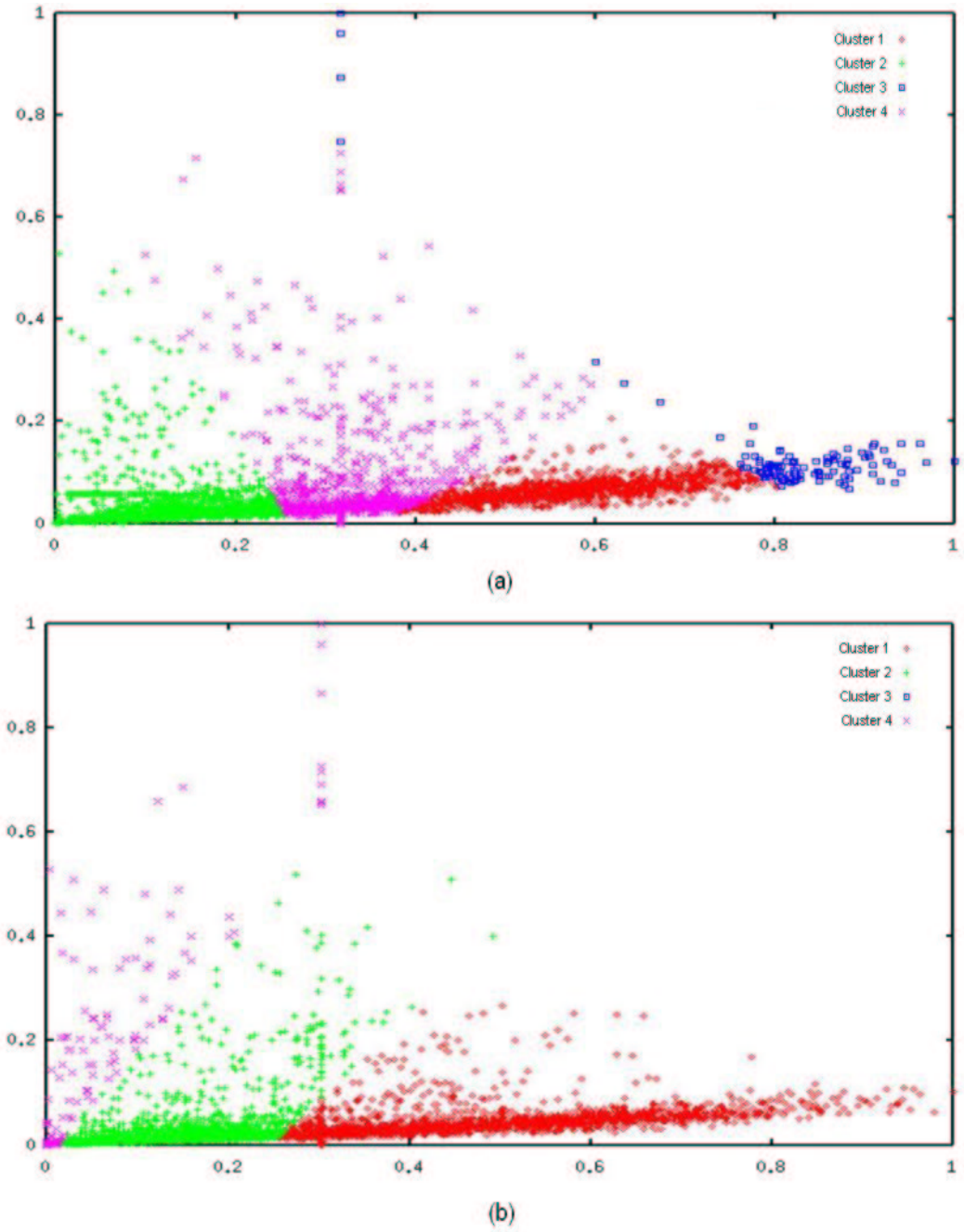

Figure 4.12: **Clusters of Sensors for weekdays and weekends.** (a) Weekdays. (b) Weekends.

Chapter 5

Related Work

There are many data analysis researchers that work on traffic data. However most of these focus on different aspects of traffic data from our work and also use a different set of traffic data. We summarize some of this work and their findings in this chapter.

O'Packi et al., [ODAB00] implement a GIS linked spatial database to keep track of components such as roads, bridges, accidents, etc. They have implemented a tool called the Transportation Information for Decision Enhancement (TIDE) that uses a Generic Query Language [GQL02] and ArcView [Arc00] interface to provide querying capabilities to the users. They provide a similar capability to their tool as ours by supporting standard queries, template queries and ad-hoc queries. However, they limit their visualizations to map visualization.

Papiernik et al., [PNCM00] implement a data warehouse to maintain and analyze performance measurements for the construction programs undertaken by Virginia Department of Transportation (VDoT).

Kwon et al., [KDPK03] have worked on archiving the data that we use for this research. They use Common Data Format to provide a universal archiving for the effective transportation of the traffic data. Nirish Dhruv and Dr. Taek Kwon [Dhr02] have also worked on building a large scale data archival system for retrieval of this data. They use the Common

Data Format to archive the data and have built applications to query this data from flat files unlike using a data warehouse approach that we have used.

Chen et al., [CPS+02] have worked on very similar data. They have implemented a freeway performance measurement system called *Performance Measurement System* (PeMS) that implements a data warehouse for single loop detector data collected from 12 districts in California. The focus of their work however is to find traffic congestion patterns [SVP03] and to make travel time predictions [PBJ+97]. They do this effectively using a web-based interface. The details of their data warehouse schema are unavailable so it is difficult to compare our methods with theirs. The use of an adaptive algorithm to determine the *g-factor* that depends on vehicle length and time of day is interesting. They use this g-factor to determine the speed of the vehicles passing the sensor.

Chen et al., [CKR+03] have also worked on detecting errors in the loop detector data and imputting values for the missing data. They have proposed a detector diagnostics algorithm that finds almost all of the bad detectors that can be detected by looking at their data visually. Their imputation algorithm takes historical information into consideration to find out relationships between the sensor and its neighboring sensors to determine the missing values more effectively than to simply interpolate the values from neighboring sensors at a given time. Their work is based on work done by Jacobson et al., [JNB90] in finding erroneous loop detectors and Nihan [Nih97].

Shekhar et al., [SLCZ00] have implemented a similar tool called *MapCube* for the same data that we use. The difference between their data warehouse and ours is that they use a concept hierarchy for space and time unlike our schema in which we split the space and time dimensions into six meaningful dimensions. Hence, their data warehouse will not facilitate queries that aggregate the data using different levels of concept hierarchy. For example, we will not be able to compare the average rush-hour and non-rush-hour traffic of weekends and weekdays in summer on the north section of I-35E with that of south section of I-35E. They find out the outliers in the data by using outlier detection algorithms to find

abnormal traffic. Their visualizations do not support swapping of dimensions to uncover interesting patterns and they have implemented only the map visualization and heat map visualization.

Chapter 6

Future Work

We can enhance our work in several ways in the future. We describe the possible improvements that can be done in all the three parts of this research.

6.1 Data Warehouse Directions

Currently, our research uses the data warehouse that has data summarized by one hour intervals. That is, each data point corresponds to the average traffic passing the sensor in an hour. We would like to reduce this interval to 5 minutes. We have already constructed such a data warehouse however have not used it for our research due to its size. We believe that doing so will help us uncover many novel patterns.

We will also like to further experiment with different softwares and optimization schemes to improve the query speeds. This will help in improving the overall efficiency of the application. Using Java Server Pages and Enterprise Java Bean technologies may also help in improving the efficiency of the overall system.

The Road Weather Information System (RWIS) has 91 sensors that record the road and weather conditions across Minnesota. There exists a correlation between traffic patterns and road and weather conditions. In the future we would like to investigate this relationship

by making use of the RWIS data and our current inductive loop data. This may help us in discovering the weather and traffic correlation and predict the traffic conditions based on weather conditions and other affecting factors. We believe that this will make the predictions much more accurate.

6.2 Data Visualization Directions

We have limited our visualizations to five different kinds of visualizations. We would like to extend this to a much larger set of visualizations. In particular, it will be useful to have a scatter graph that will show how the traffic volume and occupancy are distributed. We believe this will be very useful in showing the overall distribution of the whole or a subset of the data.

It would also be interesting to provide an API to the users so that they can write their own visualizations that can be added as plugins to the application. This is however not possible through a web-based interface, but it would be a useful feature for the standalone application.

We would like to modify our web-based interface on the basis of feedback from traffic personnel. This will help us make the application much more useful. We may also gain useful ideas regarding visualizations by doing this.

6.3 Data Mining Directions

Currently, we have implemented very simplistic data mining algorithms due to a shortage of time. It would be helpful to have a full suite of data mining algorithms implemented for the traffic data. It would be a good idea to have these applications web-based and to integrate them with the web-based query and visualization tool.

Evaluation of the patterns discovered based on real life traffic patterns will help us determine the usefulness of our data mining algorithms. Extending these tools to mine the

RWIS data will also help in discovering relationships between road-weather conditions and traffic patterns.

Chapter 7

Conclusions

There is an increasing need to analyze traffic data to solve various traffic-related problems, such as predicting traffic conditions and discovering interesting relationships in the traffic data such as the seasonal variation of traffic conditions. These relationships can be used to understand traffic conditions and perhaps to mitigate any potentially bad traffic conditions before they occur. The understanding of traffic conditions and their relationship with various factors such as day of the week, time of the day, etc. and spatial relationships is important to both commuters as well as traffic authorities. The commuters will be able to choose a route with good traffic conditions and traffic authorities will be able to mitigate bad traffic conditions by analyzing this traffic data.

The data warehouse that we have created using historical traffic data effectively addresses the problem of determining traffic patterns and discovering interesting relationships between the data and various temporal and spatial factors. We have performed simple queries to show the effectiveness of the tool. These queries produced interesting, (though not unexpected) results. For example, using a simple query we determined that the rush-hours are between 8 and 10 in the morning and between 3 and 7 in the evening. We also defined a measure called Occupancy-Volume ratio (O/V ratio) that can be used to compare the traffic conditions on various highways. We determined the O/V ratio for

all the highways and found that highway TH-212 has the best traffic conditions and highway I-694 has the worst traffic conditions. We believe that this tool coupled with domain knowledge will be very useful to discover interesting patterns in the traffic data.

Queries take a long time to execute on the data warehouse due to its size. We optimized the data warehouse using indexes to reduce the query times drastically. Our results show that there is approximately 94% reduction in the query execution times after indexing. The query times increase approximately exponentially with the addition of new dimensions before indexing. After indexing, the query execution times increase linearly with respect to the addition in the number of dimensions. This is a significant optimization.

The novel approach for designing multiple dimensions for time and space for this data warehouse facilitates ad-hoc complex queries that can uncover interesting patterns in the data. We demonstrated the use of splitting time and space dimensions into meaningful dimensions using various visualization techniques such as map visualization, bar graphs, heat maps and line graphs. These visualization techniques allow the user to understand the massive amount of traffic data by presenting it in a summarized form. The web-based interface presents a user-friendly way to query the data and view the results online from remote locations. We present multiple visualization techniques that can cater to users with different aptitudes to effectively understand the data. For example, we were able to uncover the abnormal behavior of sensors 305 and 309 on highway TH-212 during March 2002 and June 2002 respectively using map visualization, bar graphs and heat maps. This demonstrates that the visualization techniques can lead to discovery of malfunctioning sensors or anomalous traffic conditions. Using line graphs we also concluded that there exists a strong correlation between the hour of the day and the traffic conditions. The traffic conditions are similar at a given hour of the day regardless of the highways. There also exists a strong correlation between the day of the week and traffic conditions. We also discovered a strong correlation between volume and occupancy. These conclusions allow us to exploit the redundancy in the traffic data to compress this data effectively and to predict

the traffic conditions accurately.

The data mining techniques we implemented make use of the data warehouse to discover interesting relationships in the data and assist the user in predicting traffic conditions on a highway at certain point in time. The Self Organizing Maps tool characterizes the data. Using this tool we discovered that the sensors that have bad traffic conditions on weekdays have little traffic on weekends. We also discovered that about 20 to 30 percent of the sensors have bad traffic conditions during from 7am until midnight and approximately the same number of sensors have good traffic conditions on them in the same time period. Most of the sensors have low traffic on them and about 5 to 15 percent of the sensors have heavy traffic on them.

We used the Apriori algorithm to extract association rules from the data. We found rules that indicate a strong correlation between volume and occupancy. We also found rules to characterize the traffic conditions according to each day of the week. These rules show that the volume is low on weekends and the occupancy is moderate on weekdays. We further found association rules for the highways. We discovered that the if the occupancy is moderate on I-694 then the volume is low with a confidence of 98%. This allows us to conclude that the traffic conditions on highway I-694 are bad.

We performed clustering on the data to determine the clusters of the sensors. We discovered that most of the sensors have low occupancy. We also performed experiments to compare the clustering of the sensors at rush-hours to non rush-hours and discovered that there is a significant rise in the number of sensors having high volume and low occupancy during rush-hours. We also analyzed the change in the clustering from rush-hours to non rush-hours with respect to each highway and found that highway MN-62 shows a significant change in traffic conditions from rush-hours to non rush-hours. We also compared the clustering of sensors on weekdays to that on weekends and determined that the volume of sensors is lower on weekends.

All of the queries were easy to design and perform using our data warehouse. We

believe that the power to query the data warehouse directly, visualization using multiple techniques and data mining tools to discover novel patterns in the data will effectively address the needs of the commuter as well as the traffic authorities in understanding the traffic conditions and making sound decisions based on this knowledge.

Bibliography

[AdlCVT88] B. Angeniol, G. de la Croix Vaubois, and J. Le Texier. Self-organizing feature maps and the traveling salesman problem. *Neural Networks*, 1:289–293, 1988.

[AIS93] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the Association for Computing Machinery Special Interest Group on Management of Data (ACM SIGMOD) International Conference on Management of Data*, pages 207–216, Washington D.C., May 1993.

[Arc00] ArcView, Geographical Information System. http://www.esri.com/software/arcview, 2000.

[AS94] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of 20th International Conference Very Large Databases (VLDB)*, pages 487–499, Santiago, Chile, Sept 1994.

[CKR+03] C. Chen, J. Kwon, J. Rice, A. Skabardonis, and P. Variaya. Detecting errors and imputing missing data for single loop surveillance systems. *Transportation Research Record*, 2003.

[CPS+02] C. Chen, K. Petty, A. Skarbardonis, P. Varaiya, and Z. Jia. Freeway performance measurement system-mining loop detector data. *Transportation Research Record 1748*, pages 96–102, 2002.

[Dhr02] N. Dhruv. Design of large scale data archival and retrieval for transportation sensor data. Master's thesis, University of Minnesota, Duluth, July 2002.

[GCB+97] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Journal of Data Mining and Knowledge Discovery*, 1(1):29–53, 1997.

[GQL02] Generic Query Language (GQL).http://gql.sourceforge.net, 2002.

[HF95] J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. In *Proceedings of Very Large Databases Conference (VLDB) 1995*, pages 420–431, Zurich, Switzerland, 1995.

[HK01] J. Han and M. Kamber. *Data Mining: Concepts and Techniques.* Morgan Kaufmann Publishers, 340 Pine Street, Sixth Floor, San Francisco, CA 94104-3205, USA, 2001.

[HMS00] D. Hand, H. Mannila, and P. Smyth. *Principles of Data Mining.* MIT Press, Five Cambridge Center, Cambridge, MA 02142-1493, USA, 2000.

[HPYM04] J. Han, J. Pei, Y. Yin, and R. Mao. Mining frequent patterns without candidate generation. *Data Mining and Knowledge Discovery*, 8:53–87, 2004.

[HRU96] V. Harinarayan, A. Rajaraman, and J. Ullman. Implementing data cubes efficiently. In *Association of Computing Machinery Special Interest Group on Management of Data (ACM SIGMOD) 1996*, pages 205–216, 1996.

[Inm96] W. Inmon. *Building the data warehouse.* John Wiley and Sons, New York, 1996.

[JMF99] A. Jain, M. Murty, and P. Flynn. Data clustering: a review. *Association for Computing Machinery (ACM) Computing Survey*, 31(3):264–323, 1999.

[JNB90] L. Jacobson, N. Nihan, and J. Bender. Detecting erroneous loop detector data in a freeway traffic management system. *Transportation Research Record 1287*, pages 151–166, 1990.

[KDPK03] T. Kwon, N. Dhruv, S. Patwardhan, and E. Kwon. Common data format archiving of large-scale intelligent transportation systems data for efficient storage, retrieval, and portability. *Transportation Research Record 1836*, pages 111–117, 2003.

[KHC97] M. Kamber, J. Han, and J. Chiang. Metarule-guided mining of multi-dimensional association rules using data cubes. *Knowledge Discovery and Data Mining*, pages 207–210, 1997.

[Koh82] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[Koh90] T. Kohonen. The Self-Organizing map. In *Proceedings of Institute of Electical and Electronics Engineers (IEEE)*, volume 78, pages 1464–1480, 1990.

[KTS+87] T. Kohonen, K. Torkkola, M. Shozakai, J. Kangas, and O. Venta. Microprocessor implementation of a large vocabulary speech recognizer and phonetic typewriter for Finnish and Japanese. In *Proceedings of European Conference on Speech Technology*, volume 2, pages 377–380, Edinburgh, Scotland, 1987.

[MCC03] R. Maclin, D. Crouch, and C. Crouch. Northland Advanced Transportation Systems Research Laboratory (NATSRL) outreach and education program annual report. *Northland Advanced Transportation Systems Research Laboratory (NATSRL) Annual Report*, 2003.

[MGM97] I. Mumick, D. Guass, and B. Mumick. Maintenance of data cubes and summary tables in a warehouse. In *Association of Computing Machinery Special*

Interest Group on Management of Data (ACM SIGMOD) 1997, pages 100–111, 1997.

[MTV94] H. Mannila, H. Toivonen, and A. Inkeri Verkamo. Efficient algorithms for discovering association rules. In *American Association for Artificial Intelligence AAAI Workshop on Knowledge Discovery in Databases (KDD)*, pages 181–192, Seattle, Washington, 1994.

[MyS99a] The MySQL Benchmark Suite http://dev.mysql.com/doc/mysql/en/MySQL_Benchmarks.html, 1999.

[MyS99b] MySQL^{TM} http://www.mysql.com, 1999.

[Nih97] N. Nihan. Aid to determining freeway metering rates and detecting loop errors. *Journal of Transportation Engineering*, 123(6):454–458, 1997.

[ODAB00] P. O'Packi, R. Dubois, N. Armentrout, and S. Bower. Maine's approach to data warehousing for state departments of transportation. *Transportation Research Record 1719*, 1(1037):227–232, 2000.

[PBJ+97] K. Petty, P. Bickel, J. Jiang, M. Ostland, J. Rice, Y. Ritov, and F. Schoenberg. Accurate estimation of travel times from single-loop detectors. *Transportation Research Record 971043*, 1997.

[PNCM00] D. Papiernik, D. Nanda, R. Cassada, and W. Morris. Data warehouse strategy to enable performance analysis. *Transportation Research Record 1719*, 1(0603):175–183, 2000.

[SA96] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *Proceedings of Association for Computing Machinery (ACM)Special Interest Group for Management of Data (SIGMOD) 1996*, pages 1–12, 1996.

[SJ90] J. Samarabandu and O. Jakubowicz. Principles of sequential feature maps in multi-level problems. In *Proceedings of International Joint Conference on Neural Networks*, pages II–683–II–686, Washington D.C., 1990.

[SLCZ00] S. Shekhar, C. Lu, S. Chawla, and P. Zhang. Data mining and visualization of twin-cities traffic data. *Technical Report*, 2000.

[SON95] A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. In *Proceedings of the 21st Very large Databases (VLDB) Conference*, pages 432–443, Zurich, Switzerland, 1995.

[SVP03] A. Skarbardonis, P. Varaiya, and K. Petty. Measuring recurrent and non-recurrent traffic congestion. *Transportation Research Record*, 2003.

[TDR02] Traffic Data Research Laboratory.http://tdrl1.d.umn.edu, 2002.

[Val84] L. Valiant. A theory of learnable. *Communications of Association for Computing Machinery (CACM)*, 27:1134–1142, 1984.

[Val85] L. Valiant. Learning disjunctions and conjunctions. *International Joint Conference on Artificial Intelligence (ICJAI) 1985*, pages 560–565, 1985.

[Zak00] M. Zaki. Scalable algorithms for association mining. *Institute of Electrical and Electronic Engineers (IEEE) Transactions on Knowledge and Data Engineering*, 12(3):372–390, 2000.