Final Report 1999-28

MINNESOTA DEPARTMENT OF TRANSPORTATION

Research

Archival of Traffic Data: Phase II

Minnesota Guidestar®

43422876

Technical Report Documentation Page

1. Report No.	2.	3. Recipients Accession No.
MN/RC - 1999-28		

4. Title and Subtitle	5. Report Date
ARCHIVAL OF TRAFFIC DATA: Phase II	December 1998
	6.
7. Author(s) Shashi Shekhar Xinhong Tan	8. Performing Organization Report No.
9. Performing Organization Name and Address University of Minnesota Computer Science 200 Union Street, S.E. Minneapolis, Minnesota 55455	10. Project/Task/Work Unit No. 11. Contract (C) or Grant (G) No. (c) 71984 TOC # 138
12. Sponsoring Organization Name and Address Minnesota Department of Transportation 395 John Ireland Boulevard Mail Stop 330 St. Paul, Minnesota 55155	13. Type of Report and Period Covered Final Report 1995-1998 14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract (Limit: 200 words)

Traffic centers gather information from traffic sensors at regular intervals, but storing the data for future analysis becomes an issue. This report details work to improve the speed and effectiveness of traffic databases.

In this project phase, researchers redesigned the data model based on the previous phase's data model and decreased the storage requirements by one-third. Researchers developed a web-based Graphical User Interface (GUI) for users to specify the query of interest; the outcome of the performance tuning gave users reasonable response time.

The beneficiaries of this effective database would include the driving public, traffic engineers, and researchers, who are generally not familiar with the query language used in the database management system. This report summarizes the detailed reference, such as benchmark query, sample data, table schema, conversion code, and other information.

17. Document Analysis/Descriptors		18. Availability Statement
Physical design	Oracle implementation	No restrictions. Document available from:
Graphic user interface	Performance tuning	National Technical Information Services,
		Springfield, Virginia 22161

19. Security Class (this report)	20. Security Class (this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	158	

Archival of Traffic Data: Phase II

Final Report

Prepared by

Shashi Shekhar, Xinhong Tan

Computer Science Department

University of Minnesota

Minneapolis, Mn. 55455

December 1998

Submitted to

Minnesota Department of Transportation

Office of Research Services

395 John Ireland Boulevard, MS 330

St. Paul, Mn. 55155

Acknowledgments

We would like to thank Jim Aswegan, and Patt Otto at the Minnesota Department of Transportation (Mn/DOT) for their comments and patience. We would also like to thank Lowell Benson and Eil Kwon at Center of Transportation Studies of Univeristy of Minnesota for their help in the project.

Contents

List of Figures

List of Tables

Executive Summary

Traffic sensor data is gathered from traffic sensors at regular intervals, so the storage issue becomes the bottleneck of the database performance. The beneficiaries of this effective database would include the driving public, traffic engineers and researchers, who are generally not familiar with the query language used in the database management system. In this project phase, we redesigned the data model based on previous phase's data model, thus decrease the storage requirement by one third; a Web-based Graphical User Interface(GUI) was developed for users to specify the query of interest; the outcome of the performance tuning gave users pretty reasonable response time. This report summerizes our contribution on these aspects; it includes the detailed reference, such as, benchmark query, sample data, table schema, conversion code, etc.

1 Introduction

1.1 Background

The traditional method of data archival at the Traffic Management Center(TMC) has several limitations. The data is maintained for only a couple of years, and thus long-term trends cannot be analyzed. Also the data is maintained in a flat file format and it is difficult to extract an interesting set of data (e.g. the impact of a Vikings' football game on nearby traffic flow). It is not unusual for researchers to spend weeks simply extracting the relevant data sets, even though they would prefer to focus on analysis.

In 1995, the University of Minnesota, the Traffic Management Center and the Freeway Operations group started the development of a database to archive sensor network measurements from the freeway system in the Twin Cities. The sensor network includes a couple of thousand loop-detectors, which provide digital readings to TMC every 30 seconds. The preliminary design of the database is well underway. The current effort is focused on archiving of 5-minute averages and on facilitating simple queries based on traffic zones and time-lines. It is currently facing performance problems, and reports are being generated using a small subset of data. This is expected, since the data volumes are outside the reasonable performance zone of the workstation databases, as shown in Figure 1.

We have actively participated in the current Minnesota Department of Transportation(Mn/DOT) effort to archive 5-minute data through improved database design. We have also worked closely with the Minnesota Dept. of Transportation to identify a suitable design for database servers in the Travlink and Genesis projects. Thus the research team has the required background to carry out the project and transfer the technology.

The primary beneficiaries of an effective database server for an Intellegient Transportation System(ITS) data archive would include the driving public, traffic engineers and researchers. The archive will deliver the information collected by traffic sensors for the design and validation of ramp control algorithms. It will allow the freeway design group to evaluate an alternative design based on the actual measurement of traffic flow patterns on existing designs. It will also bring the data to drivers for pre-trip route planning, and to researchers for the design and validation of the ITS components.

The results from the project will benefit three related implementation projects: (i) The Traffic database for researchers in the ITS Lab. (e.g. Lowell Benson, Eil Kwon); (ii) Civil Engineering's

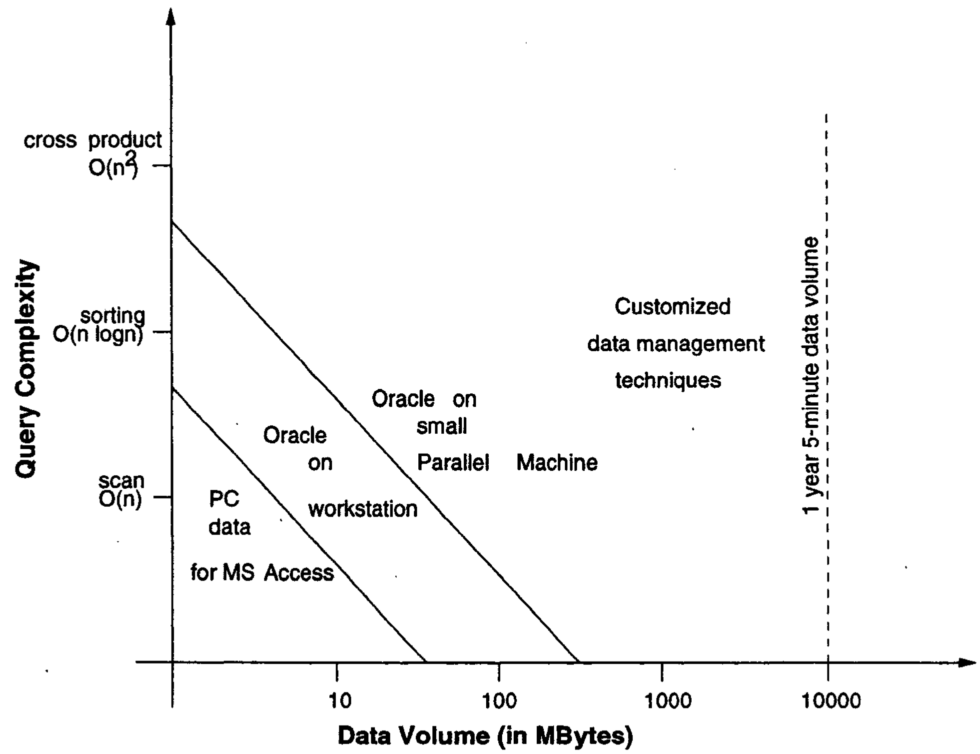

Figure 1: Reasonable Performance Zone for Databases. Query complexity denotes the expected response time as a function of the data-volume (more precisely, as a function of the number of measurements)

databases for traffic simulators (Davis, Michalopoulos), and (iii) The Mn/DOT operational TMC archive (Jim Aswegan).

1.2 Objectives

Engineers dealing with the design, construction, operation and maintenance of freeways are interested in the information that can be obtained by analyzing the sensor measurements collected at the Traffic Management Center. Thus the effective and efficient archival of sensor measurements at the TMC is critical. The archival project can provide: (a) Speed of access, (b) Assurance of data quality (e.g. integrity, consistency), (c) Automation of data collection and distribution, (d) Automation of routine data summarization and processing, (e) Integration of data from multiple sources, and (f) Communication and Management of change. Phase I of this project helped to create an archive of 5-minute data at the Traffic Operation Group at Minnesota Department of Transportation to generate summary reports. We provided conceptual design (e.g. the Entity-Relationship Model) as well as the logical design (e.g. normalized tables) in Phase I beside providing assistance in data transfer and format conversion in Phase I. Phase I revealed a performance bottleneck in archiving. The physical design issues relating to performance tuning need to be addressed in Phase II. Other issues of interest in Phase II relate to the creation of a traffic database in the ITS laboratory within Center of Transportation Studies(CTS), U of M.

The specific objectives for phase II are (a) to study physical database design techniques to address the performance bottleneck in processing 5-minute data for Mn/DOT Traffic Operations queries, (b) to develop a graphical user interface to facilitate queries on the 5-minute data, (c) to ascertain the requirements for 30 second data archiving and (d) to advise ITS traffic data personnel on Structured Query Language(SQL) level issues, assuming a properly installed Oracle database management system.

We will study physical design techniques to address the performance bottleneck in processing 5-minute data for typical queries from the Traffic Operations group of Mn/DOT. The volume of a 5-minute data for a year is beyond the reasonable performance zone for the workstation database chosen by Mn/DOT. We will study customized techniques to improve the performance of the workstation databases.

We will develop a graphical user interface (GUI) to facilitate querying the 5-minute data. The user interface acts as a front-end to the Oracle database running on Windows NT. The GUI will display the metropolitan area freeway/highway map and will allow the interactive selection of a time-range and a set of stations. The user will be able to select the granularity of the query (i.e. the level of summarization). Results will be presented as tables.

We will advise the ITS traffic database personnel on SQL level issues regarding the design and implementation of the ITS Traffic database, assuming properly installed Oracle software.

1.3 Scope

The project will focus on exploring techniques to remove bottlenecks in the 5-minute data and techniques to manage the volume of 30-second data.

1.4 Tasks

The work is divided into the following five tasks:

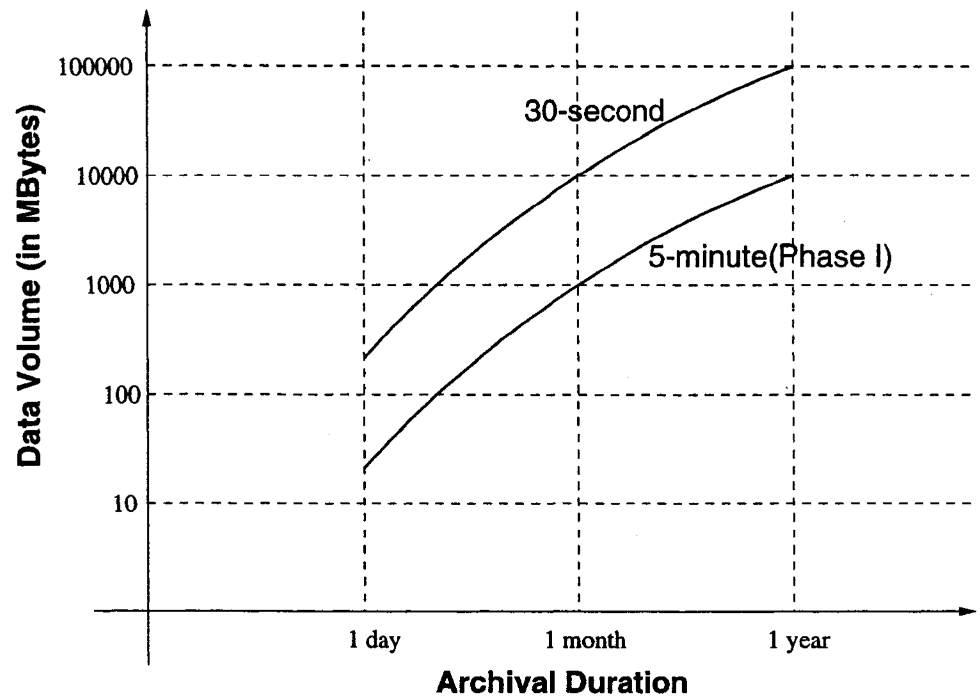

Figure 2: Data Volume Vs. Archival Duration

1.4.1 Task 1: Definition of Requirement

We will collect, then measure a set of queries and tasks to define the scope of performance tuning for the 5-minute data as it relates to query accesses. These may come from either the archival mechanism in the Computer Science laboratory or from a current data access site, either at Mn/DOT or the ITS Laboratory. We will also collect requirements for the use of 30 second data to assess the impact of data's size on the logical and physical database design.

Note: Since this task was not urgent, we dropped it and added a new task: performance tuning.

Deliverable: A 30 sec. requirement document and a document describing the bottleneck with 5 minute data

1.4.2 Task 2: Physical Design for 5-Minute Table

We will develop a simulator to take in different queries and different techniques (e.g. indexing, pre-computation, query strategy, etc.) and evaluate the performance. Techniques will be chosen to allow testing on the Mn/DOT platform, i.e. Oracle on a Personal Computer(PC). The simulator accuracy will be calibrated using the measurement results from Task 1.

Deliverable: A physical design of the database.

1.4.3 Task 3: Graphical Query Interface

We will develop a graphical interface to facilitate the selection of subsets of sensor data. Users will select a set of stations and a time interval via interaction with a visual representation of the freeway map. The interface will formulate a query to the database to retrieve the selected subset of the data.

Deliverable: Software which implements the graphical user interface to extract a subset of data related to a set of selected stations and a specified time interval. We will also provide a user manual for the software.

1.4.4 Task 4: Installation of an Oracle Traffic Database

We will jointly work with the ITS Lab Manager to install an Oracle Database Management System(DBMS) on a Windows NT platform. The ITS lab manager will be responsible for proper installation of the Oracle DBMS on a Windows NT environment in the ITS lab. We will be responsible for the implementation of the traffic database (e.g. tables, logical/physical models) in the ITS lab, given a properly installed Oracle DBMS.

Deliverable: Advice on SQL level issues relating to the operation of the database, as needed.

1.4.5 Task 5: Preparation of the Report

Generate a report documenting the results and details from tasks 1,2 and 3. Submit a results paper for potential presentation at next year's CTS conference and at a national conference.

Deliverable: A final report.

1.5 New Task: Performance Tuning

We will identify suitable optimization techniques available in Oracle to improve the performance of the traffic database, such as index creation, access methods, and the like.

Deliverable: Advice on SQL level issues relating to the performance of the database.

2 Work Delivered

2.1 Data

Sensors embedded in the freeways and interstates monitor both the occupancy and the volume of traffic on the road. At regular intervals, this information is sent to the Traffic Management Center, where it is permanently stored in a binary file. The binary file is accessed by the Center for Transportation Studies, where the data is used for research on traffic modeling and in experiments. This binary format is advantageous because the size of the file is small, 2.5 MB, and is easily read by a computer. However, the file structure is operating system dependent because of byte ordering, structure packing, and type sizes. These restrictions mean that the binary files created on a OS/2 system cannot be read on a UNIX system, forcing the data conversion program to run on a DOS/Windows machine.

```
struct {
        char szSWLevel[8];
        char end1;
        char szFLLevel[8];
        char end2;
        unsigned short ZeroRes1;
        unsigned short ZeroRes2;
        unsigned short nDetectorCnt;
        unsigned short nStructureSize;
        unsigned short nDailyRecord;
        unsigned short tDataType;
        unsigned short dYear;
        unsigned char dMonth;
        unsigned char dDay;
}       Data5MinFileHdr, *pData5MinFIleHdr;
```

Figure 3: Structure of the Header File

Description The binary file has a header containing the file's creation date and the number of detectors in the report. Pseudo-code for the header is shown in Figure 3. Sensor data in the binary file is ordered by time and detector identifier. The structure of a sensor record is shown in Figure 4 . **Vol** stores the volume for the five-minute time period and has a range of 0 to 255. **Occ** is a

```
struct {
    unsigned char Vol;
    unsigned long Occ: 10;
    unsigned short Status: 3;
    unsigned short Flag: 3;
} Int5MinData_t, *pInt5MinData_t;
```

Figure 4: Structure of the 5-Minute Detector Record

ten-bit value that stores the occupancy for the five-minute interval and that has a range of 0.0is kept by the three-bit **Status** variable. The values of this variable range from -4 to +3. The **Flag** variable is a three-bit value that stores the detector flag code. Valid values for this variable range from -4 to +3. **Flag** and **Status** variables indicate the error type, if a transmission or detector error has occurred.

From the header, the record size can easily be computed. Each record consists of an array of "nDetectorCnt" structures which are "nStructureSize" bytes each, so multiplying "nDetectorCnt" by "nStructureSize" yields the record size in bytes. The first record starts immediately after the header.

There are no time stamps on the record, so the time must be determined by the record number. The first record in the file (record 0) is always the time slice starting at midnight. For 5-minute data, there will be 288 records.

2.2 Benchmark Queries

To redesign the 5-minute table and do the performance tuning, it is important to have a proper set of benchmark queries. Fifteen queries are listed as follows. Among them, Q1-5 are provided by Dr. Eil Kwon, Q6-12 are by Jim Aswegan, and Q13-15 is from the Paradox menu interface document.

```
Q1. Get 5-min Volume, occupancy for detector ID = 10 on Oct. 1st, 1997 from 7am to 8am

Q2. Get 5-min Volume, occupancy for detector ID = 8 on Oct. 1st, 1997

Q3. Get 5-min Volume, occupancy for detector ID from 1 to 5 on Oct. 1st, 1997
from 7am to 8am

Q4. Get 5-min Volume, occupancy for station ID = 9 on Oct. 1st, 1997 from 7am to 8am
```

```
Q5. Get 5-min maximum Volume, maximum occupancy for station ID = 40 on Oct. 1st, 1997
from 6am to 7am

Q6. Get  the sum of hourly volume for station ID = 4 from Oct. 1st, 1997 to Oct. 5th
, 1997

Q7. Get 5-min average volume, maximum occupancy for station ID = 20  on Mondays
in  Oct, 1997 between 800-805, 805-810, 810-815

Q8. Get hourly  volume for station ID = 40 on Monday and Tuesdays in Oct, 1997

Q9. Get 5-min volume, occupancy  for all detectors in station ID = 24 from 7am to 8am
on Oct. 1st, 1997

Q10. Get 5-min volume, occupancy for a set of stations on highway I35W-NB
with milepoint between 0.0 and  4.0 from 7am to 7:30am on Oct. 1st, 1997

Q11. Get 5-min volume for a set of stations on highway I35W-NB between Co Rd 42  and
Burnsville Pkwy on Oct. 1st, 1997

Q12. Get the average of AM rushhour hourly volume  for a set of stations on highway I
35W-NB with milepoint between 0.0 and 4.0 from Oct. 1st, 1997 to Oct. 5th , 1997

Q13. Get hourly volume  for a set of stations in zone "1A" from 6am to 9am
on Oct. 1st, 1997

Q14. Get hourly volume  for the ramp detector  in zone "1A"   from 6am to 9am
on  Oct. 2nd, 1997
Note: the test zone is 35E south, defined by station ID 633~643

Q15. Get daily volume  for a set of stations on highway I35W-NB  with milepoint
between 0.0 and  4.0 from  Oct. 1st, to Oct 5th 1997
```

2.3 Graphical User Interface

To facilitate queries on the 5-minute data, a graphic user interface(figure 6) is provided. It is implemented in Hyper Text Makeup Language(HTML), and corresponding Common Gateway Interface(CGI) script written in PERL is used to generate the SQL statement. The generated SQL statement will be sent to the Oracle database, and the traffic data of interest will be displayed in either stack table format or metrix format.

2.4 Conversion Program

Traffic sensor data stored at the traffic Management Center is in a binary format, and the conversion program written in Visual C++ is used to convert the binary file to fixed format ASCII text file which can then be loaded into the Oracle database using the **Loader** utility in Oracle.

2.5 Creation of a Traffic Database

A test database mainly used for the benchmark study has been created in the ITS laboratory within CTS, University of Minnesota. In this database, six tables have been created and loaded with the data in table 1.

Table Name	Contents	Source
Fivemin	Oct 1 to 23, 1997 sensor data	Eil Kwon
Datetime	Oct 1 to Nov 31, 1997 time id	Automatic
Detector	3250 detector	Doug Lau
Station	770 stations	Pat Otto
Route	17 highways	Pat Otto
StatRoute	Relationship between station and route	Pat Otto
Xstreet	Relationship between xstreet and route	Jim ASwegen

Table 1: Available Tables

2.6 Reports, Meetings

During the phase II period, maintaining good communication helps the project stays on the right track.

- Reports
  - "Summary of Alternative Designs" by Anuradha Thota and Xinhong Tan in Dec 1997
  - "Project Progress" by Xinhong Tan in March 1998
- Meetings

- Jul. 1997 – Dec. 1997 Bi-weekly
- Jan. 1998 – Feb. 1998 No Meeting
- Mar. 1998 – Present Bi-weekly

3 Task 2: Physical Design of 5-minute Table

3.1 Problem Statement

Using current structure(refer to table 2), the 5-minute table alone takes 39MB approximately. With all the other reports for the detecors and stations included, approximate 76MB are required. This number would increase tremendously as the number of days increases. Loading 5-minute data into database is a slow process, so redesigning the 5-minute table is required. Since the problem of loading and querying is related to storage, the objective is to reduce storage size given the following constraints:

- Conversion effort
- Response time of query
- Future compatibility
- Derived data

3.2 Candidates

There are other four candidates other than the current structure listed in table 2, and they are described below.

- Current: In this structure, each record corresponds to one sensor data for one detector at one 5-minute interval.
- Proposed-I: Compared with current structure, where each record corresponds to one sensor data item for one detector at one 5-minute interval, in this model, one day's data for one detector is one record, which is stored in an array format. This array is made up of a sequence of volume-occupancy-validity values in order by time interval. This method compresses these values into one array, thus saving column overhead by using a single column and row overhead by storing one day's data as one record, hence reduce disk storage space
- Proposed-II: Considering the current structure, at the same time interval, we have one record for each different detector. The same readdate and time fields should be repeated in these records, and since readdate is stored in DATE format and time is in VARCHAR format, and these two fields use 11 bytes which is a little less than half of the all bytes in one row. In

Candidate	Table	Attributes	Type	Primary Key	Foreign Key	Index
Current	Fivemin	Detector	Number	Yes	–	Yes
		Readdate	Date	Yes	–	Yes—Comp
		Time	Char	Yes	–	Yes—index
		Volume	Number		–	
		Occupancy	Number		–	
		Validity	Char		–	
		Speed	Number		–	
		Dayofweek	Char		–	
Proposed-I	Fivemin_day	Readdate	Date	Yes	–	Yes—Comp
		Detector	Number	Yes	–	Yes—index
		Vol_Occ_Validity	varchar		–	
Proposed-II	Fivemin	Detector	Number	Yes	–	Yes—Comp
		Timeid	Number	Yes	Yes	Yes—index
		Volume	Number		–	
		Occupancy	Number		–	
		Validity	Char		–	
	Datetime	Timeid	Number	Yes	–	Yes
		Readdate	Date		–	
		Time	Char		–	
		Dayofweek	char		–	
Mn/DOT	Fivemin	Detector	Number	Yes	–	Yes
		Readdate	Date	Yes	–	Yes—Comp
		Hour	Number	Yes	–	Yes—index
		Dayofweek	char		–	Yes
		Vol_5_min	Number		–	
		Occ_5_min	Number		–	
		Val_5_min	Number		–	
		Vol_15_min	Number		–	
		Occ_15_min	Number		–	
		Val_15_min	Number		–	
		Vol_mn_hr	Number		–	
		Val_mn_hr	Number		–	
Binary	Fivemin	Detector	Number	Yes	–	Yes
		Readdate	Date	Yes	–	Yes—Comp
		Time	Char	Yes	–	Yes—index
		Volume	Number		–	
		Occupancy	Number		–	
		Validity	Char		–	

Table 2: Candidates for 5-minute table

the proposed-II method, we use timeid to indicate the 5-minute time interval in the original table, which will be 6 bytes if we want to store one year's data; another table called datatime is a lookup table which is used to represent the relationship between timeid and the actual data and time.

- Mn/DOT: In this structure, one hour's data for one detector is stored in one record. This record includes the fields for detector, date, hour, the day of week; it also has twelve consecutive fields for twelve 5-minute intervals in one hour, four consecutive fields for four 15-minute intervals in one hour, and one field for the current hour. Compressing the original twelve records in current structure into one record and using the time order to indicate the detailed time reduces disk storage space, and it also compacts the 15-minute and hourly information into this single record.
- Binary: The idea is very simple, because the binary format occupies less storage space than ASCII.

3.3 Methodology

Given the above five candidates, we need to know their space requirements. Then when we analyzed the four constraints, we can choose the choice of structure.

To know the candidates' space requirements, we can implement each of the candidates and obtain the numbers from the system. Since this was not practical at that time and would be very time consuming, the following methodology was used to calculate the space usage(see the below)

1. Column_size = Column_length + Column_overhead
2. Row_size = Column_size + Row_overhead
3. Usable_blockspace = Block_size - Block_overhead
4. Rows_inblock = floor(Usable_blockspace / Row_size)
5. Block_spertable = floor(Rows_intable / Rows_inblock)
6. Blocks_indatabase = Blocks_pertable + Table_overhead

Using step 1 and 2, Row_size can be calculated. By multiplying Row_size with the Rows_intable, we can obtain the number of megabytes in the table 4. Because the data in Oracle is stored in terms of blocks, the block size could be calculated using the remaining steps.

Assumption in ORACLE

- Column_length is dependent on the data type specified for the column
- Column_overhead is one byte per column
- Row_overhead is five bytes per row
- Block_overhead is at least 80 bytes
- table_overhead is one block per table
- Block_size is determined when a database is created, and the default is 2K.

The accuracy of this size prediction model is about 5% under estimation. This is attributed to the block overhead(80 bytes per 2048 bytes) in Oracle.

Structure		Column and Byte size
Current	data	detector(4) Date(7) Time(4) Vol(3) Occ(3) Val(1) Speed(4) Dayofweek(1)
	Index	detector(4) Date(7) Time(4)
Proposed-I	data	detector(4) Date(7) vol_occ_val(2016)
	Index	detector(4) Date(7)
Mn/DOT	data	detector(4) Date(7) hour(2) 5-min fields(180) 15-min fields(60) hour(15)
	Index	detector(4) Date(7) hour(2)
Proposed-II	data	fivemin table: detector(4) timeid(5) Vol(3) Occ(3) Val(1)
		datetime table: timeid(5) Date(7) Time(4) dayofweek(1)
Binary	data	detector(4) Date(7) vol_occ_val(650)
	Index	detector(4) Date(7)

Table 3: Calculation of Candidates' Space Usage

Table 3 shows the basic numbers for the calculation of space usage of these five candidates, and the six steps above show the details of the calculation.

3.4 Comparison of Candidates

A summary of the designs are given in table 4, which provides the following information:

- The storage size of the tables and the indexes of the models considered in MB and in blocks, which have been calculated using the six steps introduced above.
- The conversion effort required for the implementation of these models, which means the programming effor to convert the binary file into an ASCII file corresponding to the database structure.

Candidates		Current	Proposed-I	Proposed-II	Mn/DOT	Binary
Storage	Table	36.86	6.512	22.125	20.74	2.13
Size	Index	23.96	0.067	15.57	2.074	0.067
Per	Total MB	60.82	6.579	37.795	22.8	2.197
Day	Total Blocks	31099	6437	19263	12039	1636
Conversion Effort		No	Yes	Minimal	Yes	Yes
Effect of 30s data		No	Yes	No	Yes	No
Queries	Q1	2	4	1	1	5
	Q2	2	4	1	1	5
	Q3	2	4	1	3	5
	Q4	3	4	1	3	5
	Q5	3	4	1	4	5
Creation of Derived Data		Not very efficient	Needs PL/SQL	Not very hard	Easy	PL/SQL and host program

Table 4: Summery of Candidates

- The future compatibility(the effect of 30s data is considered), which means whether or not the structure can be used to represent 30-second data, since that is the next goal in the long term.

- A comparison of the response time of five queries, which is a rough estimate ranking the execution of the five queries based on the cost analysis in terms of disk access. The bigger the number, the slower the query.

- The effort required to obtain the derived data, since the 5-minute table is the core table, and the Mn/DOT and CTS researchers also need information on fifteen minute, hourly and daily tables. Some effort is required to provide availability from the core 5-minute table, different model will make a difference on the effort needed.

From the analysis in table 4, we conclude:

- The current model occupies a large disk space. This is seen as a negative factor; in spite of having reasonable response time for most queries

- Proposed-I model occupies less disk space, but the query statement needs Procedure Language(PL)/SQL and is not very straightforward.

- In the Mn/DOT model, the conversion effort is substantial. The fifteenmin and hourly data exist in the table when it is created, and no special effort is needed to create them. However, some effort is needed for the creation of station data. The queries need to be bounded by an hour, so queries using a fraction of hour might take a longer time.
- In the proposed-II model the conversion effort required is minimal. Derived data need to be created, but all the queries could be run efficiently compared to other models.
- The binary model takes very little space. The queries need a host program and PL/SQL to be transformed to a readable format.

3.5 Conclusions/Choices

The Mn/DOT model and proposed-II could be the two recommended models for the final structure.

- When conversion effort is considered, the Mn/DOT model needs a new loading program. Proposed-II needs little modification to the existing loading process, but requires the creation of a new table, but this table takes little time and effort.
- Considering future compatibility, using the Mn/DOT model increases the number of columns, which if using proposed-II the same format can be retained.
- For the derived data, in the Mn/DOT model the fifteenmin and hourly data are loaded in the table at the same time the fivemin data is loaded. Effort is needed only for the creation of station data. In proposed-II this data has to be derived.
- Query is more flexible in the proposed-II than the Mn/DOT model.

The final choice on the structure of 5-minute table is proposed-II, which was decided upon in one of meetings.

4 Task 3: Graphical User Interface(GUI) design

4.1 GUI-Graphic Design

4.1.1 Problem Statement, Evaluation Criteria

Problem Statement Given the traffic database with a well-designed structure (refer to Proposed-II in table 4), and with Mn/DOT/CTS researchers and traffic engineers with little knowledge of SQL as users, we need to develop a graphic user interface to facilitate users' needs for data retrieval from this traffic database within the constraints of the given schema.

Evaluation Criteria The following are three criteria used for the GUI design:

- It must be easily used by Mn/DOT/CTS researchers as non-Computer Science users, e.g. we need to avoid complex part of SQL, like GROUP BY;
- It need not be equivalent to SQL, and should be a subset of SQL;
- It should facilitate a set of benchmark queries on traffic data;

4.1.2 Candidates

We have four candidates, which are described as follows:

Candidate I: Data Flow Query Language(DFQL)

In DFQL, a data-flow diagram screen first shows users the available tables in this database and the algebric operators on tables, as well as the join operation between two tables. Then a series of screens will appear to let users fill in the required conditions under the guidance of the full knowledge of SQL statement for the desired query. For example, there are eight screens used to define a benchmark query Q1(refer to appendix A) as follows:

1. Data flow diagram with $\sigma, \pi, \bowtie$ tables;
2. Selection condition on detector table;
3. Join condition for detector and 5-minute tables;
4. Join condition result of 3 and datetime;

5. Selection condition on datetime table;

6. Join;

7. Project on Volume, Occupancy;

8. File/output specification;

Pros: Good for researchers and not for traffic engineers.

Cons: Too close to SQL and too detailed.

Candidate II: Current GUI in Paradox

Figure 5 is the diagram for Paradox Menu. In this screen, there are pull-down menus and fill-in boxes for query specification. The screen can be roughly divided into three parts:

Output : It is on the upper left part of this screen. The user can define the query report type, and choose the output style from two alternatives: tabular and crosstab by day, decide the result either ordered by time or location, and specify that the result is for detector 5-minute, or detector hourly, or detector daily or station 5-minute, or station hourly or station daily.

Time : It is on the lower left part of this screen. User can fill in the begin date and time and end date and time for the period of interest, and they also can specify Rush time interval by clicking on one of two buttons: am Rush and pm Rush. Since the traffic pattern will be different on a weekend than on a workday, so user can also choose one from Type of Days.

Location : It is on the lower right part of this screen. The users can identify the specific highway segment of interest by filling in the highway name, and the begin and end milepoint. Further restriction can be applied by choosing the specific speed and volume of interest, or filling in the zone of interest. The user may be interested in the detectors on one specific lane, and they can choose one from Lane pull-down menus.

Pros: No requirement of any SQL knowledge.

Cons: Limited function of defining queries which are needed by researchers, for example, lack of cross street specification and statistical analysis.

Candidate III: Proposed Custom GUI

The figure 6 is the diagram for Custom GUI. This GUI prototype was developed using HTML, and the reason for choosing HyperText Makeup Language(HTML) is that it is quite easy to use and

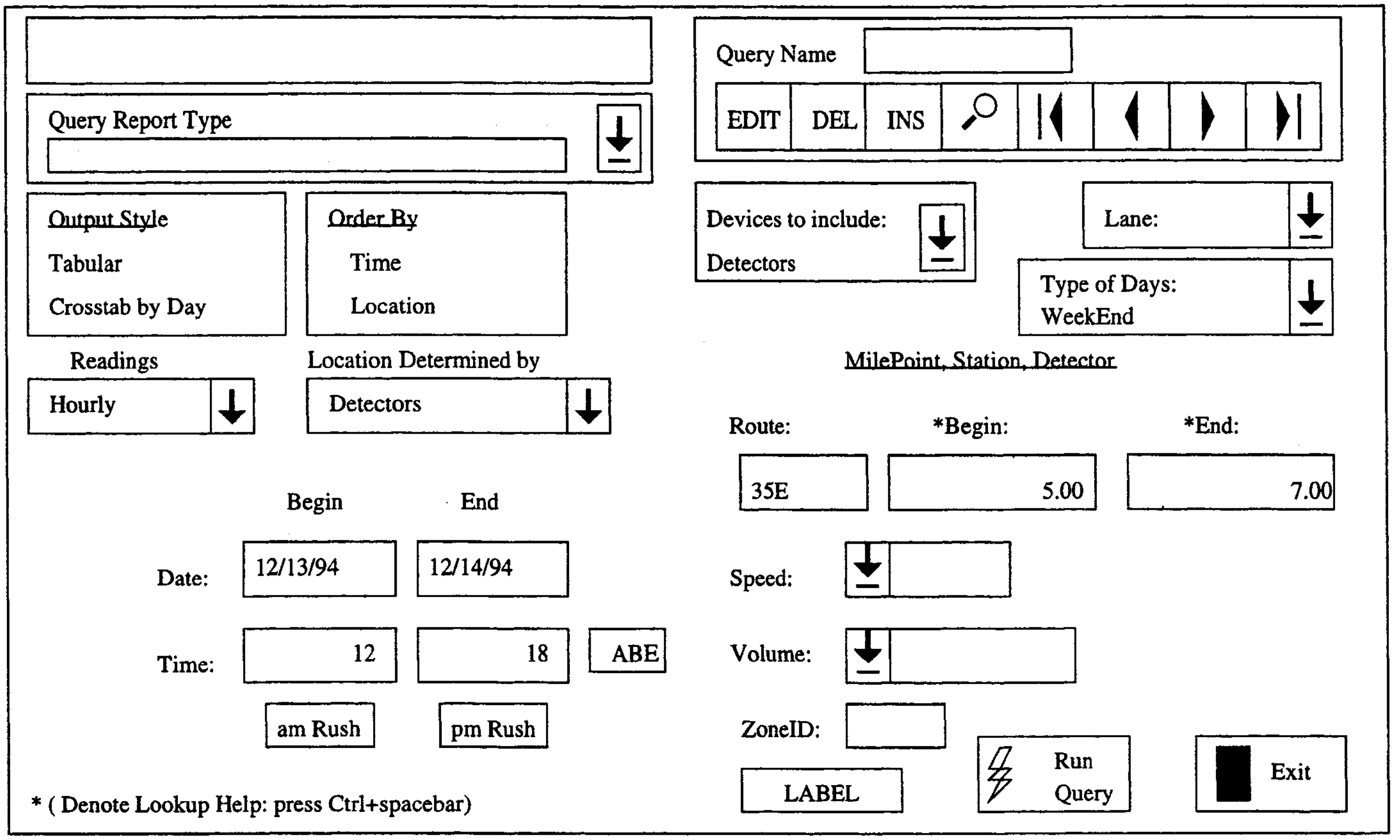

Figure 5: Paradox Menu Screen

quick to implement. Since it is a prototype, we can implement it in other ways, depending on the software support in the ITS laboratory and on executive requirements. As far as the long-term trend for the use of traffic data are concerned, the Web-based approach will be suitable.

In this prototype(figure 6), the screen is organized from up to down into four parts: geographic location, time interval, desired sensor data level and desired attributes. A detailed explanation for each part will follow.

Geographic location : There are two ways to specify the geographic location for the queried sensor: *sensor location*, and *ID input.*

For *sensor location*, two alternatives are available: highway and ramp metering zone, In the highway specification, first user choose one of highways of interest from a list of highways, and lane choice from the lane list(currently unavalible due to the lack of support data), then identify the specific segment of one highway via two different ways, one is to choose the from reference point and to reference point, and the other is to choose the from cross street and to cross street. In the pre-defined zone, one zone name is chosen from the zone list(currently one fictitious name created

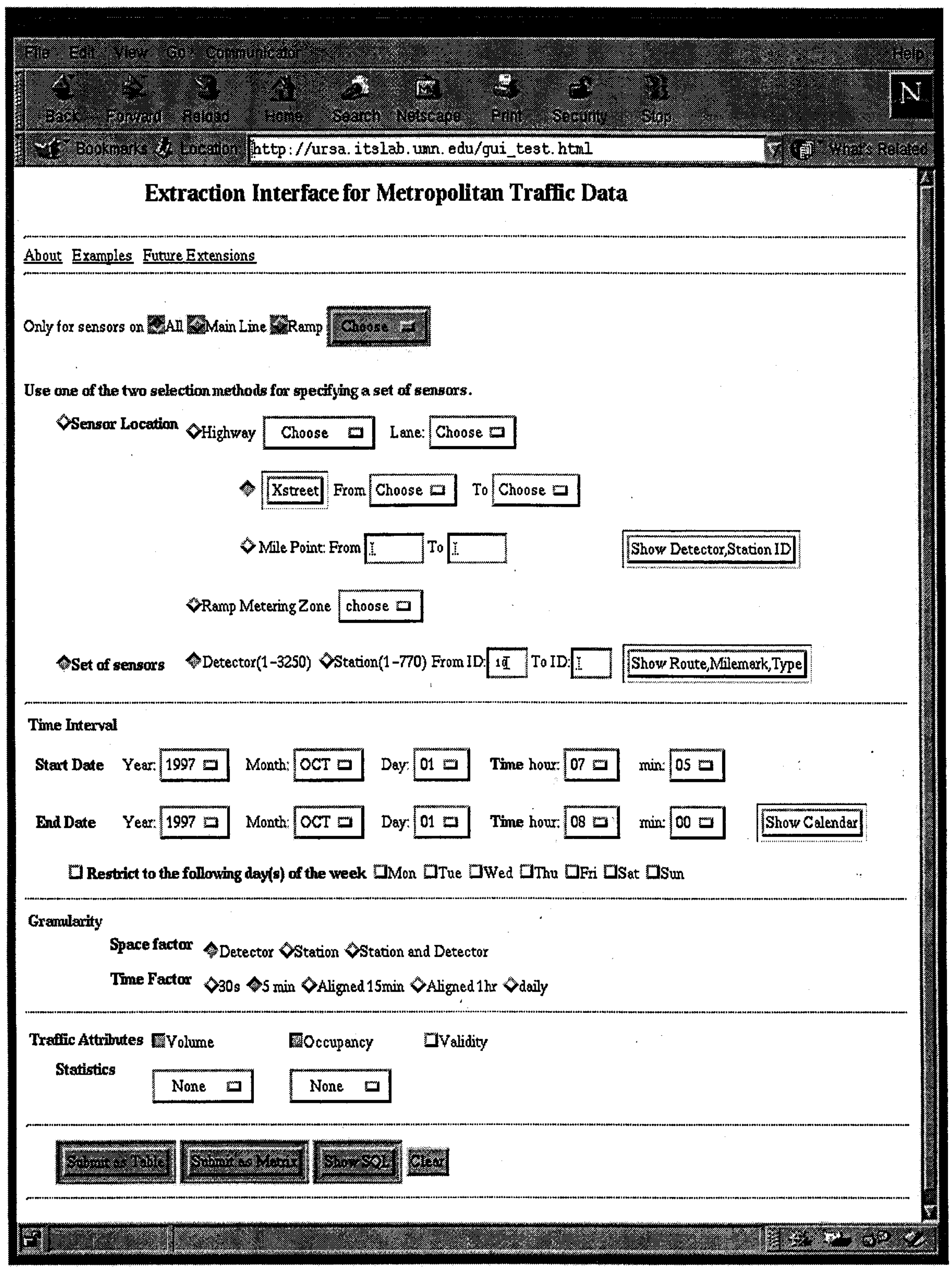

Figure 6: GUI window

for the station from 633 to 643).

For *ID input*, either enter one specific detector/station ID, or give a range of detector/station ID to identify the sensor(s) of interest.

Time interval: There are lists of Year, Month, Day, Hour and Minute for start date and end date. Depending on the time interval of interest, two or more lists will be specified. Since the traffic information is related to different days in a week, another specification is included for which day of interest.

Desired sensor data level: The level is one both space and time. In the space factor, either detector or station is clicked to indicate that the desired data is on detector or station level; In the time factor, one of 30 seconds, 5-minute, aligned 15-minute,aligned 1 hour or daily can be chosen.

Desired attributes: Volume, Occupancy, and validity are three attributes, and one or more can be chosen. For volume, one of four kinds of statistics can be specified, and for occupancy, one of three kinds of statistics can be specified.

After specifying these four parts, the user has two choice for output dispaly, either "submit as table" or "submit as metrix", then the retrieved data appears on a new sceen with the specified format. Another button "show SQL" is used to dispaly the generated SQL statement mainly for the debug purpose. "clear" botton is for the refresh of the screen. **Pros:** A well organized screen, easy to use, no requirement of SQL knowledge.
Cons: Limited function for defining queries which have restriction on volume.

Candidate IV: Augumented Proposed GUI(Candidate III + Map)

Figure 6 with the first part which is changed to the figure 7 is the screen for augumented Proposed GUI(Candidate III + Map)(will be implemented in the followup project): In this interface, the map part is used to specify the geographic location in the query by clicking the spot of interest, and the other part is the same as in the candidate III. After the specification is done, the form is submitted and the query result can also be shown graphically in the map part.
Pros: A very straightforward way of specifying location in a query and result representation.
Cons: Need sufficient knowledge on map, and more effort is needed for implementation.

4.1.3 Methbodology

User testing with test queries and interface:

- Let the user define the query in each interface;

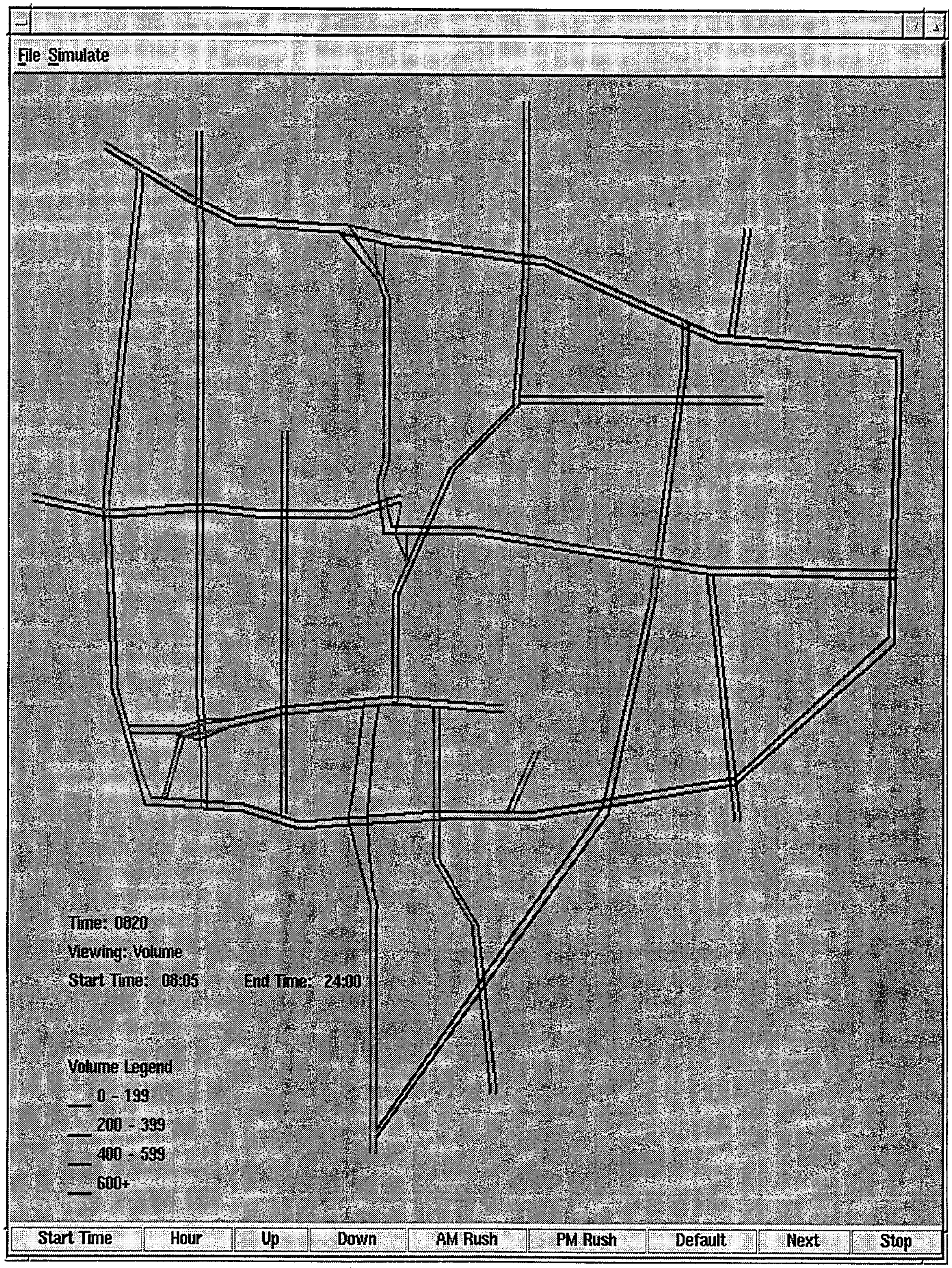

Figure 7: Map-Based Screen

- Listen to user feedback;

4.1.4 Comparison of Candidates

The summary of the interface design is in table 5, and the following information is given in this table:

Criteria			DFQL	Current GUI	Proposed GUI	GIS-Based
Expressive Power		Q1	Yes	Yes	Yes	Depends
		Q4	Yes	No	Yes	Yes
		Q5	Yes	No	Yes	Yes
		Q12	Yes	Yes	No	No
		Q15	Yes	No	Yes	Depends
		SQL Equivalence	= SQL	Subset	Subset	Subset
Easy to Use	Assumed Knowledge	SQL	Yes	No	No	No
		Traffic Data	Yes	Yes	Yes	Yes
		Map	No	No	No	Yes
		mile point	Yes	Yes	Yes	Yes
	Potential Complexity	Group By	Yes	N/A	No	N/A
		Join(inner/outer)	Yes	No	No	No
		Table structure(key)	Yes	No	No	No
		Nested Query	Yes	No	No	No
What is visible on top screen?			Join Condition Table definition	Pulldown menu Highway Name Street Name	Highway Name Street Name	
Effect of changes on table definition on user			A lot	Little	Little	Little

Table 5: Summary of Comparison Result

- The expressive power of the candidates: this means whether or not the queries which the user needs can be formalized via the interface. Here we take five benchmark queries as examples to indicate the expressive power of the candidates. The other aspect is their equivalence to SQL, which indicates whether or not the steps to specify the query are the same as those in the SQL statement. The more they are equivalent to SQL, the more expressive power they have, since we need to formalize the user query into a SQL statement.
- Estimation of interface usefulness in terms of two factors:

- Assumed knowledge to use the interface, which means what kind of knowledge the user is required to know before using the interface, and the relevant knowledge includes:
  * SQL, which stands for Standard Query Language, which is a compressive database language, and has statements for data definition, query and query. For non-CS people, it is not easy to learn.
  * Traffic Data, the meaning of volume, occupancy and validity and what kind of information can be exacted from these data.
  * Map, which means familiarity with the highway and the definition of mile points
- Potential Complexity of query specification in terms of the following aspects related to SQL level issues:
  * Group By, which is needed when we want to apply the aggregate function, such as maxmium, minimum, average and the like. The *group by* clause specifies the grouping attributes, and it is error prone.
  * Join(inner/outer), which is needed when we query more than two tables, and it is really the case in this database. If the join is executed as a nested loop, the specification of inner and outer table will have a great effect on the performance which can be optimized.
  * Table structure(key), which is the table definition, index specification and the integrity and reference constraints.
  * Nested Query, which is used to specify a relatively complex query

• The visibility of features needed in the formation of a query, which means how many things the user needs to remember, such as the join condition, table definition, highway name and the gazeteer name.

• The effect on users of changes in table definition, which means that whether or not the user needs to know about these changes, and how much effort users need to adjus exert to the new table definitions.

In table 5, for expressive power, "yes" indicates the interface can formalize the query, and "no" means it can not; for ease of use, "yes" means that the interface needs the assumed knowledge or has the potential complexity, "no" means it does not.

From the analysis of table 5, the following conclusion was obtained:

- DFQL has the most expressive power, but it requires knowledge of SQL, which is really hard for a Mn/DOT/CTS researcher. It also has the most potential complexity, and a change of table definition will produce considerable effect on users. It has moderate visibility, which is not really interesting to the users

- The current GUI has less expressive power than DFQL, but it does not need SQL, map and gazeteer knowledge; only traffic data knowledge is needed. It does not have potential complexity or more visibility. There will be little effect on users in case of changes in table definition

- The proposed GUI has less expressive power than DFQL, and more than the current GUI. In term of other aspects, it is almost similiar to the current GUI, except that more knowledge of the gazeteer is needed and has a little less visibility.

- Augumented Proposed GUI has least visibility among these four candidates, and in terms of ease of use, it is similiar to the proposed GUI. As for expressive power, it really depends on the implementation level of the map.

4.1.5 Conclusions/Choices

The proposed GUI is the recommended interface and in the long term, Map + Proposed GUI is the recommended interface:

- As far as expressive power is concerned, the proposed GUI is moderately capable of speicifying the queries, while the augumented Proposed GUI is implementation-dependent

- In terms of ease of use, these two are similiar

- Considering visibility, the proposed GUI has more than the augumented proposed GUI, and thus more effort is needed from users

- In case of changes in table definition, both of them will have little effect on users

Interfacing to the rest of the system

The following figure 8 shows one of apporaches:

- Defining a query

- Structure DFQL/SQL
- Parameter: GUI – form/Map

- Datatool for interoperality

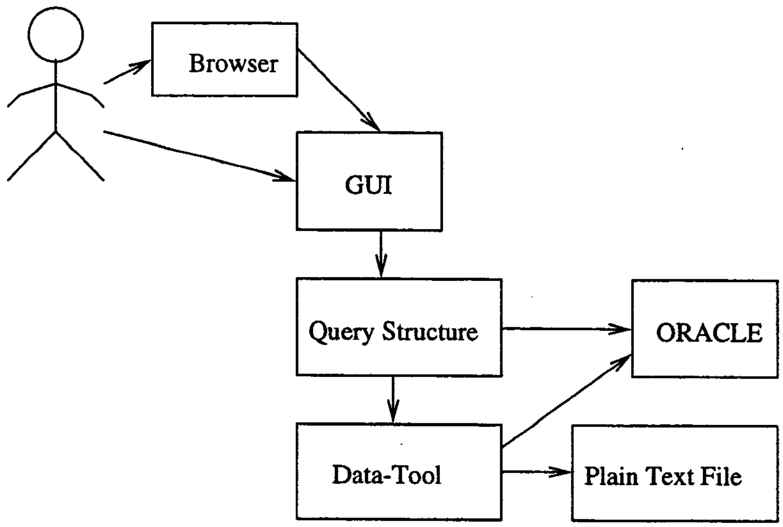

Figure 8: Graphic User Interface

4.2 GUI-Generating SQL Query

4.2.1 Problem Statement

With a specific GUI form for data extraction and the traffic database schema, as well as the user input for the desired query, a corresponding SQL query should be generated, and the following constraints are considered:

- Correctness: which means that the form of the SQL statement is correct and the result of the SQL statement is what the user wants.
- Optimal usage of tables: which means that the choice of tables used in the SQL statement should be proper: not too much, which will cause problems in performance; and not too few, which would not accomplish the user query.
- Faster response time: which is always what the user expects.

4.2.2 Candidates

```
SELECT ReadDate, Time
FROM Fivemin, Datetime
WHERE detector BETWEEN 4 and 6
and volume > 5
and detector.timeid = datetime.timeid
```

Figure 9: Example SQL statement for the generation of SQL

```
SELECT ReadDate, Time
FROM Fivemin, Datetime, Detector, station, Route, Statroute
WHERE timeid IN
        (SELECT timeid
        FROM Fivemin
        WHERE detector = 4
        or detector = 5
        or detector = 6
        and volume > 5)
```

Figure 10: Example SQL statement for the generation of SQL

To generate an SQL query, three aspects will be considered, which are Query Form, Restrictions and Set of Tables, and each of them has two or three candidates. The explanation of these aspects as well as their candidates will be given as follows:

1. **Query Form:** which is related to the flexibility of the SQL statement, that is, one query can be expressed in several different forms, for example, figures 9 and 10 are two different forms of the query "Get the data, time when detector 4 or 5 or 6 has a volume larger than 5". Roughly speaking, the query forms can be categorized into two kinds: Flat SQL and Nested SQL.

   - Flat SQL: In this form, the statement has only one SELECT clause, one FROM clause, and one WHERE clause; optionally it has one GROUP BY clause and one ORDER BY clause, And it does not have IN, EXISTS, NOT EXISTS. One example form is given in figure 9, and this form is easier ORACLE to optimize unless a nested query yields a very small result set.
   - Nested SQL: In this form, the statement has more than one of these clauses, and it needs IN, EXISTS, NOT EXISTS. One example form is given in figure 9, and contrary to the flat SQL, it is usually harder to optimize.

2. **Restrictions:** the factors in terms of performance and optomization issues. In the SQL statement, we have a set of operators which are used to express the user's query condition in the WHERE clause, and different operators have different implementations, thus hasing different effects on performance and optimization. Based on the query needs in this traffic database, the following three operators will be discussed:

   - the use of "<", ">" in the *where* clause: as a logic operator, it can be used to specify some restriction on attributes, like volume > 5, and the combination of "<" and ">" can be used to identify the range for the attributes, for example, to specify one time period, readdate < "Oct-01-97" and readdate > "Oct-04-97", or to specify one highway segment, mp < 10 and mp > 1. One example form is given in figure 9.
   - the use of "BETWEEN" in the *where* clause, which is usually used to represent the range; for example, readdate BETWEEN "Oct-01-97" AND "Oct-04-97". It has less expressive power than "<", ">", but it has the best performance for a given primary or secondary index, e.g. time, detector. One example form is given in figure 9

- the use of "or" in the *where* clause: as a logical connective, it is used to express a disjunctive condition and it is much harder to process and optimize than AND, which is used to express a conjunctive condition. One example form is given in figure 10

3. **Set of Tables** the table choice in the FROM clause of one SQL statement, since user input does not contain this information, and it should be inferred from user input. Thus we can have the following two extreme choices:

   - A universal set of tables: which means that we do not need to make a choice. We just put all of the available tables into the FROM clause; that is, 5-minute, Datetime, Detector, station, Route, or Statroute. One example form is given in figure 10. This way will save time in generatiing SQL, but it has a side effect on the performance.
   - the least set of tables required: which means that we just use 5-minute, Datetime, which has the best performance for simple query on one or a set of detectors. But for a query on the station level or geographical location specification by highway, we need the nested SQL, which is hard to optimize. One example form is given in figure 9.

4.2.3 Methodology

Given these candidates and their aspects, different combinations could be obtained, which would affect the performance, but we did not go into the deteails in comparing these combinations. An informal selection is used to make our choice, that is:

- the flat query
- Using a minimal number of tables
- Using the BETWEEN operator for the attributes with an index specified

After we determine the representation of the SQL statement, we need to develop an algorithm which can translate user input into the SQL statement with the given form. The following figure 11 is the SQL Statement that we want, table 6 shows the input variables from GUI, and table 7 shows the input values from GUI for these input variables.

In table 6, we can see that these input variables can be roughly grouped into four kinds: Geographic location, Time interval, Desired sensor data level, and Desired attributes. In each kind, we have several levels of input, the detailes are referred to in table 6. For these input variables,

```
SELECT    <attribute list1>
FROM      <table list>
WHERE     <conditions>
GROUP BY  <attribute list2>
```

Figure 11: The format of the SQL statement

most are provided with option values which are listed in the table 7, and a few variables need to be entered.

The psuedocode for the CGI which is used to generate the SQL is listed in Appendix B. The following is the brief description for this psuedocode:

Given the input values which are a part of table 7, the first four arrays are defined to store the four clauses in the SQL statement, which are attributeAtoms for the SELECT clause, tableAtoms for the FROM clause, whereatoms for the WHERE clause, and groupbyAtoms for the GROUP BY clause. Then procedure table_list(I) is used to add items to tableAtoms, procedure attribute_list(I) is used to add items to attributeAtoms, procedure where_clause(I) is used to add items to whereatoms, and procedure GroupBy_list(I) is used to add items to groupbyAtoms; finally, using JOIN to concatenate each item in each of the four arrays into a string with different separators, these four strings are concatenated into the SQL query.

Test Cases

The following are two test cases for the generation of a query.

```
Q1: Get 5-min Volume, Occupancy for detector "5" on Oct. 01, 97 from 6am to 7am
INPUT: time1 = 5min, attrib1 = volume, attrib2 = occupancy, volume_column = none
       occ_column = none, space1 = detector, sensor1 = detector, start_year = 97
       start_month = oct, start_day = 1, end_year = 97, end_month = oct,
       end_day = 1, start_HOUR = 06, start_minute = 00, end_hour = 07,
       end_minute = 00
OUTPUT:
  SELECT
  ReadDate, Time, volume, occupancy, xtan.fivemin.detector
  FROM xtan.DateTime, xtan.fivemin
  WHERE ReadDate = '01-OCT-97'
  AND Time BETWEEN '0605' AND '0700'
  AND xtan.fivemin.Detector = 5
  AND xtan.fivemin.timeid = xtan.DateTime.timeid
```

<table>
<tr><th colspan="4">Option Part</th><th>Input Part</th></tr>
<tr><td rowspan="5">Geographic location (sensor_select)</td><td rowspan="3">sensor location (gazateer_select)</td><td rowspan="2">Highway (highway)</td><td>Ref Point (startend_select)</td><td>startmp, endmp</td></tr>
<tr><td>Xstreet (startend_select)</td><td>street1, street2</td></tr>
<tr><td colspan="2">Predefined Zone (zone_list)</td><td></td></tr>
<tr><td rowspan="2">ID Input (sensor1)</td><td colspan="2">Detector</td><td>id</td></tr>
<tr><td colspan="2">Station</td><td>id</td></tr>
<tr><td rowspan="3">Time Interval</td><td colspan="3">Start Date (start_year, month, day, hour, minute)</td><td></td></tr>
<tr><td colspan="3">End Date (end_year, month, day, hour, minute)</td><td></td></tr>
<tr><td colspan="3">Days (d1)</td><td></td></tr>
<tr><td rowspan="2">Desired Sensor Data Level</td><td colspan="3">Space Factor (space1)</td><td></td></tr>
<tr><td colspan="3">Time Factor (time1)</td><td></td></tr>
<tr><td rowspan="2">Desired Attributes</td><td colspan="3">Attributes (attrib1, attrib2,attrib3)</td><td></td></tr>
<tr><td colspan="3">Statictics (volume_column, occ_column)</td><td></td></tr>
</table>

Table 6: Input for CGI script from GUI

```
Q2: Get 5-min maximum(Volume, Occupancy ) for detector "5"  on Oct. 01, 97
from 6am to 7am
INPUT: time1 = 5min, attrib1 = volume, attrib2 = occupancy, volume_column = max
       occ_column = max, space1 = detector, sensor1 = detector, start_year = 97
       start_month = oct, start_day = 1, end_year = 97, end_month = oct,
       end_day = 1, start_hour = 06, start_minute = 00, end_hour = 07,
       end_minute = 00
OUTPUT:
   SELECT
   max(volume), max(occupancy), station
   FROM xtan.DateTime, xtan.fivemin
   WHERE ReadDate = '01-OCT-97'
```

Variable	Values
sensor_select	gazateer, sensor
gazateer_select	route, zone, parfile
highway	35W, 35E, ...
startend_select	mp, xstreet
zone_list	8A, ...
parfile_list	......
sensor1	detector, station
start(end)_year	96, 97, 98, ...
start(end)_month	1,2,..., 12
start(end)_day	1,2,..., 31
start(end)_hour	1,2,..., 24
start(end)_minute	00, 05,..., 55
d1	mon, tues
space1	detector, station
time1	5min, 15min,...,
attrib1, attrib2,attrib3	volume, occupancy, validity
volume_column, occ_column	max, min, ave, none

Table 7: Input Value for CGI script from GUI

```
AND Time BETWEEN '0605' AND '0700'
AND xtan.fivemin.Detector = 5
AND xtan.fivemin.timeid = xtan.DateTime.timeid
```

4.2.4 Comparison of Candidates

A summary of the designs is specified in table 8, which gives the following information:

- Performance: the effect of the candidates on response time, including their difference between the candidates.
- Optimization effort: which means that different query forms for the same query will require a different effort to optimize.
- Expressive Power: which means the ability to specify the query.

By analyzing of table 8, the following conclusion has been obtained:

Aspects	Candidates	Performance	Optimization effort	Expressive Power
SQL form	Flat SQL	quick	easy	low
	Nested SQL	slow	hard	high
Restrictions	<, >	slow	N/A	high
	BETWEEN	fast	N/A	low
	or	slow	hard	N/A
Set of Tables	Universal	slow	hard	same
	Lease	quick	hard	same

Table 8: Summery of Candidates for SQL statement

- Among the two different query forms, flat SQL can be easily optimized. Though it has less expressive power than nested SQL, it is sufficient for user queries.

- For the choice of operators, for the attributes with the specified index, BETWEEN will be used, and <, > and "or" will depend on the query need.

- For the choice of tables, neither of them is good, we will use a minimal set of tables without using nested SQL

4.3 Implementation Details

4.3.1 Connection between Web and Oracle

To access the traffic data which is stored in Oracle via the Web-based GUI, we need to establish the connection between Web and Oracle. There are several ways to achieve this purpose, such as, PowerBuilder, Visual basic, Java, Perl,etc.. Currently we use perl to implement the connection.

The key parts of the connection establishment is DBI, DBD::ORACLE, and CGI, and we will give each of them brief introduction. DBI, the Database Interface for perl5, is a database-independent interface for database connectivity which abstracts the complexity of understanding the low-level 'guts' of database technologies away from the programmer. DBD::ORACLE, is the ORACLE interface for perl5. The DBD implement the methods defined in DBI, eg, connect(), in a database-specific way, that is, ORACLEi way. DBI essentially acts as a conduit for the DBD modules. The programmer of applications will never even know the DBD is there! All they will be aware of is the database-independent methods defined by DBI. CGI is a simple protocol that can be used to communicate between Web forms and your program. A CGI script can be written in any language that can read STDIN, write to STDOUT, and read environment variables, i.e. virtually any programming language, including C, Perl, or even shell scripting.

The Web-based GUI is a form to be filled out by users, once users submit this form, the Web server will process these data in a CGI script, first creating the SQL statement, then making connection to ORACLE and sending query, then get the data back from Oracle, and finally displaying the result in a HTML file.

4.3.2 New features

In this section, the following five features are described in more detail to give users better understanding as well as more efficient usage.

Cross-Street

This feature provides one very useful means of specifying some highway segment which is deliminated by cross street name since cross street name is more direct to the users than the reference points. In the Web-based GUI, after users choose the highway which they are interested in, they click on the button "Xstreet", the corresponding cross street name list will appear for the From and To choice pull-down menu. In the cross street name list, each item includes the cross street name with the corresponding reference point, and in the From choice, the list is in the acsending order on reference point, while in the To choice, the list is in the descending order.

Show Detector,Station ID

One button "Show Detector, station ID" is added in the *sensor location.* This button is added for the reference purpose, and when the users specify the highway segment which they are interested in, they may also want to know which detector and station are in such highway segment before they go further to retrieve the sensor data.

Show Route,Milemark,Type

One button "Show Route, milemark, Type" is added in *ID input.* This button is also dded for the reference purpose, and when the users specify the detector(s) or station(s) which they are interested in, they may also want to know where the(se) detector(s) or station(s) are in with respect to highway before they go further to retrieve the sensor data.

Show Calendar

As we know that some traffic pattern may vary by the different days of the week, and thus we provide the choice for which day(s) of interest. In this way, it is very helpful to let user check the calendar given the specific month. That is the reason that one button "Show Calendar" is added.

Submit as Table And Submit as Metrix

Considering the different requirements on the output data format, currently two kinds of format choice are available, that is, table and metrix. The table format means that the rows grow in the order of detector/station and time interval, and columns is stable with the attribute(s) of interest; while in the metrix format, the columns grow in the order of detectors/stations, and the rows is stable with the time interval; the most important feature is that when the users want the detector and station information together, the station will be followed by these detectors which are made up of this station, such way gives the user very convenient view to analysis the traffic data.

5 Task 4: Oracle Implementation

5.1 Currently Implemented Data Model

The following is the currently implemented data model in figure 12.

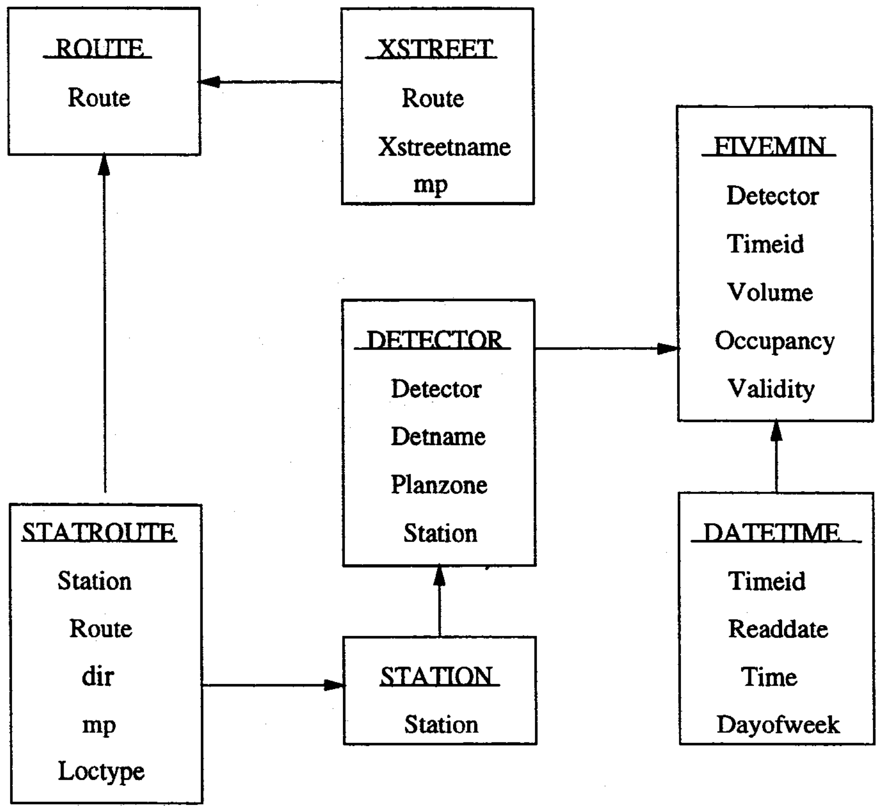

Figure 12: Data Model

5.2 Creation of a New Database

The following are some basic steps for seting up a new database:

1. Obtain from analysts, designers, and/or user representatives background information on the number of users, their usage pattern, and the expected increase in the number of users.

2. Check with the system administrater to make sure there is adequate memory and disk space. It is useful to estimate the amount of storage required to hold the data for the application system. Make sure that the disk space are contiguous. Determine the names and locations of the database files, the control file, and the redo log files.

3. Create the INIT · ORA parameter file and set the most efficient values for the parameters.

4. Create the database and tablespaces with their default storage allocation.
5. Create the users who own the tables, indexes, sequences, views, and clusters.
6. Create tables, indexes, sequences, views, and clusters.
7. Load the data.
8. Create other users.

In the above steps, each has more or less effect on the performance, and we will consider two steps below, the tablespace and the data loading.

5.3 Considerations of Tablespace

Tablespaces are logical divisions of the database. A tablespace may contain one or more database files. When a table, index, cluster, or rollback segment is created, it will be created in a tablespace with the specified storage parameters or default storage parameters for the tablespace. The number, size and location of these parts considerably affect the performance, so we recommend the following instructions:

- Tables that are commonly used together should be ideally be split across multiple tablespaces. In the test database, one tablespace is created for the 5-minute table, and one tablespace is created for the geographic location tables, such as detector, station, statroute.
- Because indexes and tables are often inserted into and read from simultaneously, splitting tables and indexes into separate tablespaces minimizes disk-head movement and allows concurrent access. From performance tuning, we can see the need for an index on 5-minutes, so one separate tablespace is created specific for the index of the 5-minute table.
- One or more temporary tablespaces are created for temporary segments. All uses must have this tablespace defined as their temporary tablespace. This tablespace must be at least the size of the largest table. This is very important because we need this space to do the temporary sorting process in the query. Two tablespaces are created for the test database.
- When tables, indexes, and rollback segments are created, they are assigned an initial storage allocaton. If that allocation is exceeded, Oracle must assign additional extents in a process called dynamic extension. Access to data is more efficient if extents are contiguous. When

a table's extents are discontiguous, access is much slower because the system needs to scan these discontiguous areas. This leads to the issue of how to specify storage parameters. In this traffic database, these geographic location tables are small and rather static, so it's quite easy to contain them in one initial extent without worrying about dynamic extension. For the 5-minute table, we need to deal with the dynamic extension problem.

Tablespace Name	Space Size	Table contained
FWFM1D	800M	5-minute table
FWFM1I	600M	5-minute table index
FWGL1D	50M	Geographic location tables
FWGL1I	50M	Geographic location table indexes
FWTP1D	800M	Temporary space

Table 9: Tablespace setup for TEST database

Table 9 gives the tablespace setup for the TEST database , which is designed for one-month's worth data.

5.4 Table Loading

In ORACLE, there is a utility called Loader to load the external data into the database. SQL*Loader requires two types of input: the external data, which can reside on disk, and control information, which describes the characteristics of the input data and the tables and columns to load. The outputs, some of which are optional, include Oracle table(s), log file, bad file(s), and discard file(s). In this database, the external data is the ASCII file which is converted from the binary file, and because it is fixed format, the control file is quite easy to write. In the following we discuss how to load the 5-minute table and other tables respectively.

In current data model, there are seven tables. Among them, tables station and route can be derived from other tables, tables detector, xstreet and statrdwy should be loaded from external data, and table fivemin needs to be loaded dynamically.

It is a trival job to load the tables detector, xstreet and statrdwy, given the fixed format text file.

The focus of loading is on the 5-minute table, because it is the core table in the traffic database, it needs to be converted from binary to ASCII, and its volume is large and will increase with days go. In the following, we will talk about the loading problem from the 5-minute table.

5.4.1 Problem Statement

Given the 5-minute table structure, and binary daily file from ftp and on-line data source, we need to develop a loading schema with the following constraints:

- Data availability in terms of the database application;
- Quick execution time
- Efficient operation

5.4.2 Candidates

There are three issues involved in the design of the 5-minute table loading scheme:

- Loading method
- Loading cycle
- Loading implementation

The explanation of these three issues and their candidates will be given as follows:

1. Loading method: this means that the job is done manually or via script. We have two tasks involved in the loading:

   (a) Get the binary file either on File Transfer protocol(FTP) or from Data Distribution Server(DDS), and

   (b) Convert the binary file to text file and load into Oracle.

   Thus, we can have four kinds of combination as shown in table 10(where "M" indicates manual and "S" indicates script):

Get binary file	M	M	S	S
Convert to text file and load	M	S	M	S

Table 10: Candidates for 5-minute table's loading methods

   The second combination will be our choices.

2. Loading cycle: this means how often the loading task should be done, and we could do it in three different cycles:

   - once a week: it can be used when the FTP site is down or other failures happen;
   - once a day: it is preferred since it can provide daily data availability;
   - every five minute(on-line): it is not practical because the loading process will keep the database system busy and the data is not accessible.

3. Loading implementation: this means how to deal with the overhead from the recovery and index. We have the following candidates given Window-based scheme, for example, one month's data available:

   (a) Load one day into Oracle via inserting and delete one day, then create new time ID in table datetime;

   (b) Recycle time ids with insert and delete after certain period;

   (c) Recycle time ids with update, and unless we add new detector in table detector, there is no insert and delete;

   (d) Drop index, bulk load one day, delete one day and then create index;

   (e) Divide the one month's data into several tables, for example, one table for one day, bulk load one day, and delete one day by just dropping one table, then use view to combine them together.

5.4.3 Comparison of Candidates

It is easy to make the choice of loading method and loading cycle, while for the choice of loading implementation, we need to compare the tradeoff on the following aspects in table 11 .

Tradeoff	Recovery Overhead			Index Overhead		Creation of Index
	Insert One Day	Delete One Day	Update One Day	Insert One Day	Delete One Day	
a	+	+		+	+	
b	+	+		+	+	
c			+			
d		+				+
e						+

Table 11: Comparison of 5-minute table's loading Implementation

Table 11 gives the following information:

- Recovery overhead: it means what should be done to restore the database when some failure occurs during the process of updating the data: either inserting or deleting or updating. For one day's data, we have 3250 $\times$ 288 rows of records, we need to identify the exact row after the failure, and it will be a time-consuming task, and should be done manually.

- Index overhead: it means what should be done with the index during the process of updating the data, either inserting or deleting. Once some failure happens, we should drop the index for the whole table and recreate the index.

- Creation of index: it means the effort needed for the creation of index for the whole table. In our experiment, we need 1 minute to create index for a table with one day's table, and 3 minutes for a table with seven days' data. Compared to the recovery overhead, this task is much less.

In the table 11, "+" means that the overhead exists, and empty cell means the overhead does not exist.

From the analysis of table 11, the candidate c and e seem practical:

- The main concern with candidate c is the tedious procedure;

- The main concern with candidate e is the its effect on the performance;

To test the candidate e's effect on the performance, we did the following experiment: five tables are created, each of them is loaded with one day's data and index is created for each table, then a view is created from these five tables, which is V_FIVEMIN in table 14.

Loading of the 5-minute Table

There are two ways to load the 5-minute table: one is to load the data with the index, and the other is to load the data without the index.

In our test, to load one day's data without index needs 0.33 hour or so, and to load one day's data with index needs 1.66 hour or so.

If we load the 5-minute data without the index, we need to create the index before we use this table. In our test, the time for the index creation on the one day, two days, three days, five days, and seven days' data are 1, 1,5, 1,5, 3, 3 minutes respectively.

From this trend, we can see that it is better to load the data without the index first, and then to create the index later.

6 New task: Performance Tuning

6.1 Problem Statement

Problem Statement Given tables and a set of queries, select index and other performance mechanisms available in ORACLE to get the best average response time.

Options for performance tuning in ORACLE:

- Index
- Star Join
- Others, like memory, I/O and contention tuning

Oracle Tool for Performance Tuning:

- **EXPLAIN PLAN:** It enables the DBA to pass a SQL statement through the Oracle optimizer and to learn how the statement will be executed by the database –the *execution plan.* That way, it is possible to learn whether the database is peforming as expected–for example, whether it uses an index on a table intead of scanning the entire database table.
- **SQL*Trace and TKPROF:** It reveals the quantitative numbers behind the SQL execution. In addition to an execution plan, SQL*Trace generates factors such as CPU and disk resources. This is often considered a lower-level view of how a database query is performing, because it shows factors at both the operating system and RDBMS levels. These numbers are stored in the trace file, and TKPROF is used to convert this file into a readable format.
- **Dynamic Performance(V$) Tables:** The V$ tables are views on the Oracle X$ tables, which are System Global Area (SGA)-held memory structures created by the database at startup. These tables and their views are updated in real time as the database runs, and provides the DBA a good view of the current status of the database.

Assumptions: fixed hardware, fixed table design.

6.2 Index

Index provides a powerful way to speed up the retrieval of data from an Oracle database, but this advantage does not come for free: there is a cost for storage space, and it also requires maintenance. The number of indexes and choice of index columns should be made carefully. In this traffic database, a 5-minute table is the key table as well as the largest table, while the rest of tables are small, so we will consider the index issue on the 5-minute table.

6.2.1 Candidates

In this traffic database, when we want to retrieve some data, we usually need to specify at least the time interval of interest, and in most cases, we also need to identify the set of sensors of interest in a specific geographic location; thus for the 5-minute table, we have three alternatives for the index:

- No index
- Cluster index: If the records of a file are physically ordered on a non-key field that does not have a distinct value for each record, that field is called a clustering field. A cluster index is one which is created for the clustering field to speed up the retrieval of records that have the same value for the clustering field. Since the field does not have one entry for each record, it may take less space. In this database, we can have cluster index on timeid, and the following is the SQL statement:

```
i). create cluster first
    CREATE CLUSTER fivemin_timeid (timeid number(6))
      PCTUSED 80
      PCTFREE 5;
ii) create table in the cluster

    CREATE TABLE C_FIVEMIN (
     ..........
     )
    CLUSTER fivemin_timeid (timeid);
iii). create a cluster index
    CREATE INDEX fivemin_timeid_ind
      ON CLUSTER fivemin_timeid
      INITRANS 2
      MAXTRANS 5
      PCTFREE 5;
```

- Primary index: If the records of a file are physically ordered on a key field that have a distinct value for each record, that field is called ordering key field. A primary index is an index specified on the ordering key field, and it has one entry for each record. In this database, we can have a cluster index on timeid, and the following is the SQL statement:

```
CREATE INDEX p_fmtimedet_i
 ON fivemin(timeid, detector)
```

6.2.2 Methodology

Three aspects are considered:

- Loading time: As we know, one of the many challenges DBAs face today is the problem of migrating data from external sources into an ORACLE database. Loader is a very versatile tool in ORACLE that loads external data onto ORACLE database tables. Upon execution, SQL*Loader creates a log file containing detailed information about the load which includes the timestamp for its execution and elapse time and CPU time, where we can obtain the loading time record.

- Space usage: The index has the time and space tradeoff, so we need to look at the space usage. This information is stored in the system table sys.dbs_tables. Using the following SQL statement will give you the number of blocks and the number of rows for the table.

```
SELECT blocks, num_rows, table_name
FROM sys.dba_tables
```

- Response time: We need to know how fast we can get by using different index choices. We can use the tool SQL*Trace and TKPROF mentioned above to measure it. The following are the steps to do it:

  1. ALTER SESSION SET sql_trace = 'TRUE'
  2. Run the query
  3. Find the latest trace file(.trc) in the directory:/ORANT/RDBMS73/TRACE/
  4. Run TKPROF as follows: TKPROF73 ***.trc ***.log
  5. View the ***.log file and find the statistics:

```
{query statement}
--------------------------------------------------------------
call      count  cpu   elapsed  disk  query  current   rows
......    .....  ...   .......  ....  .....  .......   ....
Parse         1  0.02  0.02        0      0        0      0
Execute       1  0.00  0.00        0      0        0      0
Fetch         4  0.03  0.03        1     20       10     50
......    .....  ...   .......  ....  .....  .......   ....
total         6  0.05  0.05        1     20       10     50
--------------------------------------------------------------
Misses in library cache during parse: 0
Misses in library cache during execute: 1
Optimizer hint: CHOOSE
Parsing user id: 22 (MERLIN)
```

For the response time measurement, we are interested in part of this statistic. CPU means the cpu time for all parses, executes and fetches in one-hundredths of seconds; elap means the elapsed time for these operations in one-hundredths of seconds. Parse is the statistics for the parse step performed by SQL statements; Execute is the statistic for the execute step performed by the SQL statements, UPDATE, DELETE,and INSERT statements show the number of rows processed here; Fetch is the statistic for the fetch step performed by SQL statements, SELECT statements show the number of rows processed here.

6.2.3 Comparison of Candidates

Table 12 gives the load time record for SEVen successive days' data, table 13 shows the space usage, and table 14 gives the respon se time for the fifteen benchqueries on the different 5-minute tables. C_Fivemin is the table with the cluster index, P_Fivemin is the table with the index on timeid and detector, the N_Fivemin is the table without any index.

Table Name		C_Fivemin	P_Fivemin		N_Fivemin
			Data	Index	
Load Record	First	1.33hour	0.33hour	1min	0.33hour
	Second	1.33hour	0.33hour	1.5min	0.33hour
	Third	1.33hour	0.33hour	1.5min	0.33hour
	Fourth	1.33hour	0.33hour		0.33hour
	Fifth	1.33hour	0.33hour	3min	0.33hour
	Sixth	1.33hour	0.33hour		0.33hour
	Seventh	1.33hour	0.33hour	3min	0.33hour

Table 12: Load time for the 5-minute table

Table Name	C_Fivemin		P_Fivemin		N_Fivemin
	Data	Index	Data	Index	
Num_Rows			6551600	6562847	6552007
Blocks			71659	70173	71629
MB			147	143.7	147
MB/day			21	19.5	21

Table 13: Space usage for 5-minute table

6.2.4 Conclusion

- In terms of loading time, using primary index does not need much more extra time for the creation of an index than without an index. This seems sublinear increase, and cluster index loading requires much more time;
- Considering space usage, stotage without index needs the least space, with a cluster index is the second, and with a primary index uses space almost as double as without an index.
- As far as the response time is concerned, loading with a primary index has the best performance among the fifteen queries.

The choice of index is really a time and space tradeoff, and since the hardware is becoming cheaper and cheaper, the primary index is the choice.

Query Number		C_Fivemin	P_Fivemin	N_Fivemin	V_Fivemin
		CPU ELAP	CPU ELAP	CPU ELAP	CPU ELAP
Q1	Parse	0.02 0.08	0.01 0.02	0.01 0.01	0.01 0.01
	Exec	0.00 0.00	0.00 0.00	0.00 0.00	0.00 0.00
	Fetch	0.41 1.21	0.13 0.33	26.65 46.97	0.03 0.08
Q2	Parse	0.01 0.02	0.01 0.01	0.00 0.01	0.02 0.02
	Exec	0.00 0.00	0.00 0.00	0.00 0.00	0.00 0.00
	Fetch	31.05 50.52	0.80 4.91	28.36 47.67	0.30 4.45
Q3	Parse	0.02 0.02	0.02 0.02	0.01 0.02	0.05 0.05
	Exec	0.00 0.00	0.00 0.00	0.00 0.00	0.00 0.00
	Fetch	0.42 0.54	0.12 0.47	35.16 56.64	0.08 0.11
Q4	Parse	0.02 0.05	0.02 0.02	0.02 0.05	0.05 0.05
	Exec	0.00 0.00	0.00 0.00	0.00 0.00	0.00 0.00
	Fetch	0.16 0.57	0.16 0.76	443.88 747.26	0.10 0.24
Q5	Parse	0.02 0.02	0.01 0.01	0.02 0.02	0.03 0.03
	Exec	0.00 0.00	0.00 0.00	0.00 0.00	0.00 0.00
	Fetch	0.42 0.81	0.17 0.23	36.90 57.71	0.04 0.25
Q6	Parse	0.05 0.05	very quick	0.04 0.03	0.08 0.08
	Exec	0.00 0.00		0.00 0.00	0.00 0.00
	Fetch	2.76 4.69		2575.97 4528.23	0.19 0.82
Q8	Parse	0.02 0.04	0.02 0.02	0.03 0.03	0.09 0.11
	Exec	0.30 0.37	0.49 0.61	0.00 0.00	0.00 0.00
	Fetch	367.66 1102.07	1.18 8.13	33.24 54.17	74.41 82.98
Q9	Parse	0.01 0.01	0.01 0.01	0.03 0.03	0.04 0.04
	Exec	0.23 0.23	0.00 0.00	0.00 0.00	0.30 0.37
	Fetch	194.08 932.69	38.51	33.92 63.13	331.65 743.39
Q10	Parse	0.01 0.01	0.02 0.02	0.01 0.04	0.04 0.04
	Exec	0.17 0.22	0.00 0.00	0.00 0.00	0.30 0.37
	Fetch	200.71 865.12	1.74 20.64	33.34 54.47	231.65 743.39
Q11	Parse	0.04 0.09	0.02 0.02	0.01 0.04	0.03 0.04
	Exec	0.00 0.00	0.12 0.12	0.00 0.00	0.09 0.10
	Fetch	14.11 18.99	69.19 79.22	69.95 111.64	55.60 87.33
Q12	Parse	0.03 0.05	0.03 0.03	0.03 0.03	0.05 0.05
	Exec	0.34 0.39	0.20 0.20	0.00 0.00	0.19 0.20
	Fetch	92.95 203.88	41.43 59.67	32.70 53.81	42.37 88.77
Q13	Parse	0.02 0.02	0.01 0.03	0.01 0.01	0.03 0.03
	Exec	0.20 0.23	0.26 0.28	0.16 0.21	0.21 0.25
	Fetch	126.56 205.44	129.51 179.03	118.86 162.95	77.32 99.11
Q14	Parse	0.02 0.02	0.03 0.03	0.02 0.02	0.03 0.03
	Exec	0.36 0.37	0.06 0.06	0.10 0.10	0.06 0.09
	Fetch	279.01 1073.26	9.02 38.87	35.89 63.89	19.46 48.66
Q15	Parse	0.06 0.06	0.05 0.05	0.05 0.05	0.05 0.05
	Exec	0.00 0.00	0.00 0.00	0.00 0.00	0.00 0.00
	Fetch	115.01 322.48	3.40 42.13	128.65 242.58	673.22 1576.58

Table 14: Response time for queries using the 5-minute table with different index choices

6.3 Star Join

We can see from the fifteen benchmark queries that they are characterized by one large table, that is, the 5-minute table, and some small tables like datetime, detector, statroute which join with the large table. This kind of query is called star join, and ORACLE has some internal mechanisms to optimize star join. In our test, we run some queries with star join option and without it, and the following table 15 gives the response time.

Query #	Without Star	With Star
	CPU ELAP	CPU ELAP
Q1	0.02 0.08	0.01 0.02
	0.00 0.00	0.00 0.00
	0.15 0.89	0.13 0.33
Q3	0.00 0.01	0.02 0.02
	0.18 0.21	0.00 0.00
	2.82 4.28	0.12 0.47

Table 15: Response time for queries with and w/o star join

6.4 Other Optimizations

At the database level, there are three kinds of tuning:

- Memory tuning
- I/O tuning
- Contention tuning

Each kind of tuning has a distinct set of areas that the DBA must examine. Memory tuning deals with optimizing the numerous caches, buffers, and shared pools that reside in memory and compose the core memory structures for the Oracle RDBMS. I/O tuning is concerned with maxmizing the speed and efficiency with which the RDBMS accesses the physical data files that make up its basic storage units. Contention tuning seeks to resolve problems in which the database fights against itself for database resources.

APPENDIX A

View Creation

```
create or replace view v_ftmin(detector, time, volume,occupancy, readdate,dayofweek)
as
    select /*+ STAR */
    xtan.detector.detector, substr(xtan.datetime.time,1,2)||trunc(substr(time,3,2)/15
,0) * 15, sum(volume), avg(occupancy),readdate, to_char(readdate,'D')
    from xtan.p_fivemin, xtan.datetime, xtan.detector
    where xtan.p_fivemin.timeid = xtan.DateTime.timeid
    AND xtan.p_fivemin.detector = xtan.detector.detector
    group by xtan.detector.detector, readdate, substr(xtan.datetime.time,1,2),trunc(s
ubstr(time,3,2)/15,0) * 15

create or replace view v_hour(detector, hour, volume,occupancy, readdate,dayofweek) a
s
    select /*+ STAR */
    xtan.detector.detector, (trunc((substr(xtan.datetime.time,1,2)*12 + substr(xtan.d
atetime.time,3,2)/5 + 11) / 12, 0) - 1), sum(volume), avg(occupancy),readdate, to_cha
r(readdate,'D')
    from xtan.p_fivemin, xtan.datetime, xtan.detector
    where xtan.p_fivemin.timeid = xtan.DateTime.timeid
    AND xtan.p_fivemin.detector = xtan.detector.detector
    group by xtan.detector.detector, readdate, trunc((substr(xtan.datetime.time,1,2)*
12 + substr(xtan.datetime.time,3,2)/5 + 11) / 12, 0)

create or replace view v_daily(station, volume,occupancy, readdate,dayofweek) as
    select /*+ STAR */
    xtan.detector.detector, sum(volume),avg(occupancy), readdate, to_char(readdate,'D
')
    from xtan.p_fivemin, xtan.datetime, xtan.detector
    where xtan.p_fivemin.timeid = xtan.DateTime.timeid
    AND xtan.p_fivemin.detector = xtan.detector.detector
    group by xtan.detector.detector, readdate

create or replace view v_stat_ftmin(station,time , volume, occupancy, readdate, dayof
week) as
select /*+ STAR */
station, substr(xtan.datetime.time,1,2)||trunc(substr(time,3,2)/15,0) * 15, sum(volum
e),avg(occupancy), readdate,to_char(readdate,'D')
from xtan.fivemin, xtan.datetime, xtan.detector
where xtan.fivemin.timeid = xtan.DateTime.timeid
AND xtan.fivemin.detector = xtan.detector.detector
group by xtan.detector.station, readdate, substr(xtan.datetime.time,1,2),trunc(substr
(time,3,2)/15,0) * 15


create or replace view v_stat_ftmin(station,hour, volume, occupancy, readdate, dayofw
eek) as
select /*+ STAR */
station, (trunc((substr(xtan.datetime.time,1,2)*12 + substr(xtan.datetime.time,3,2)/5
 + 11) / 12, 0) - 1), sum(volume),avg(occupancy), readdate,to_char(readdate,'D')
from xtan.fivemin, xtan.datetime, xtan.detector
where xtan.fivemin.timeid = xtan.DateTime.timeid
AND xtan.fivemin.detector = xtan.detector.detector
group by xtan.detector.station, readdate, trunc((substr(xtan.datetime.time,1,2)*12 +
substr(xtan.datetime.time,3,2)/5 + 11) / 12, 0)

create or replace view v_stat_daily(station, volume,occupancy, readdate,dayofweek) as
    select /*+ STAR */
    station, sum(volume),avg(occupancy), readdate,to_char(readdate,'D')
    from xtan.fivemin, xtan.datetime, xtan.detector
```

```
where xtan.fivemin.timeid = xtan.DateTime.timeid
AND xtan.fivemin.detector = xtan.detector.detector
group by xtan.detector.station, readdate
```

APPENDIX B

CGI Psedocode

```
INPUT: a list of values(I) for the variable listed in the table~\ref{GUIvalue}
which corresponds to a user query condition
OUTPUT: a SQL statement which is equivalent to the user query
Mehtod:
whereatoms: array of atoms for where clause, e.g. "date > start date"
tableAtoms: array of atoms for from clause e.g. Detector, DateTime,,...
attributeAtoms: array of atoms for select clause e.g. Detector.detector,
groupbyAtoms: array of atoms for group by clause e.g. Detector.detector, ...

1. Using the following procedure to form table list which has the minimal set
DateTime and fivemin
procedure table_list(I)
begin
  if space1 is equal to detector and sensor_select is not equal to sensor
        and sensor1 is not equal to detector
     add table "detector" to tableAtoms
  if gazeteer_select is equal to route
     add table "statrdwy" to tableAtoms
end
2. Using the following procedure to form attribute list which has the minimal set
ReadDate, Time
procedure attribute_list(I)
begin
  if space1 is equal to detector
     add "detector" to attributeAtoms
  if space1 is equal to station
     add "station" to attributeAtoms
  if attrib1(2) is not null and volume_column(occ_column) is equal to none
     add "volume(occupancy)"  to attributeAtoms
  if attrib3 is not null
     add "validity"  to attributeAtoms
end
3. Using the following procedure to form Where clause which has the minimal set
join condition, date restriction
procedure where_clause(I)
begin
  concatenate start(end)\_year,start(end)\_month,start(end)\_day to form
  start(end)date;
  add "readdate BETWEEN startdate AND enddate" to whereatoms;
  concatenate start(end)\_hour,start(end)\_minute to form start(end)time;
  add "time BETWEEN starttime AND endtime" to whereatoms;
  if sensor_select is equal to sensor
     if sensor1 is equal to detector
        add "fivemin\.detector = id" to whereatoms
     if sensor1 is equal to station
        add "detector\.station = id" to whereatoms
  if gazeteer_select is equal to route
     add "statrdwy.route = 35W" to whereatoms
     if startend_select is equal to mp
        add "statrdwy\.mp $ >= $ startmp" to whereatoms

  if space1 is equal to detector
     if sensor_select is equal to sensor and sensor1 is equal to station
        add "fivemin\.detector = detector\.detector" to whereatoms
     if sensor_select is equal to gazeteer and gazeteer_select is equal to route
        add "fivemin\.detector = detector\.detector" to whereatoms
        add "detector\.station = statrdwy\.station" to whereatoms
end
4. Using the following procedure to form GroupBy list
```

```
procedure GroupBy_list(I)
begin
  if volume_column is not equal to none or occ_column is not equal to none
     if space1 is equal to detector
        add "detector" to groupbyAtoms
     if space1 is equal to station
     add "station" to groupbyAtoms
end
5. Create SQL statement
procedure SQL_creation(whereatoms,tableAtoms,attributeAtoms,groupbyAtoms)
begin
  join whereatoms into a string using seperator AND;
  join tableAtoms into a string using seperator comma;
  join attributeAtoms into a string using seperator comma;
  join groupbyAtoms into a string using seperator comma;
  concatenate strings to create the SQL query
```

APPENDIX C

Oracle Table Schema and Sample Data

```
-------------------------------------------------------
FW5MD__T.SQL
PURPOSE: SCRIPT USED TO CREATE FREEWAY TABLE - FIVEMIN
         FOR PRODUCTION.
         FIVE MINUTE LOOP DETECTOR COUNTS BY DETECTOR.

CHANGE HISTORY
DATE        BY          PURPOSE
----------- ----------- -------------------------------
01-AUG-1996 G. GUNELSON INITIAL CREATION FOR PRODUCTION.
10-MAR-1998 XINHONG TAN TEST CREATION
-------------------------------------------------------
CREATE NEW TABLE - FIVEMIN
-------------------------------------------------------
*/
CREATE TABLE fivemin
 (
  detector                                  NUMBER(4)  ,
  timeid                                    NUMBER(5)  ,
  volume                                    NUMBER(3)  ,
  occupancy                                 NUMBER(4,1),
  validity                                  VARCHAR(1)
 )
TABLESPACE fwfm1d
PCTFREE 2^M
STORAGE (INITIAL     10M
         NEXT        10M
         MINEXTENTS    2
         MAXEXTENTS  121
         PCTINCREASE   0
        )
/*
-------------------------------------------------------
FWDETE_T.SQL
PURPOSE: SCRIPT USED TO CREATE FREEWAY TABLE - DETECTOR
         FOR PRODUCTION.

CHANGE HISTORY
DATE        BY          PURPOSE
----------- ----------- -------------------------------
01-AUG-1996 G. GUNELSON INITIAL CREATION FOR PRODUCTION.
10-MAR-1998 XINHONG TAN TEST CREATION
19-APR-1998 XINHONG TAN BENCHMARK TEST CREATION WITH CHANGE
-------------------------------------------------------
*/
CREATE TABLE detector
 (detector                                  NUMBER(4),
  detname                                   VARCHAR2(16),
  zone                                      CHAR(2),
  station                                   NUMBER(5)
 )
TABLESPACE fwgl1d
PCTFREE 10
STORAGE (INITIAL       100K
         NEXT          100K
         MINEXTENTS    1
         MAXEXTENTS  121
         PCTINCREASE   0
        )
```

```
/*
--------------------------------------------------------
FWRDWY_T.SQL
PURPOSE: SCRIPT USED TO CREATE FREEWAY TABLE - ROADWAY
         FOR PRODUCTION.
         ROADWAY INFORMATION TABLE.  PRIMARILY A SUPPORT
         LOOKUP TABLE.

CHANGE HISTORY
DATE        BY          PURPOSE
----------- ----------- -------------------------------
01-AUG-1996 G. GUNELSON INITIAL CREATION FOR PRODUCTION.
10-MAR-1998 XINHONG TAN TEST CREATION
--------------------------------------------------------
*/
CREATE TABLE roadway
 (route                              VARCHAR(8)
 )
TABLESPACE fwglld
PCTFREE 10
STORAGE (INITIAL       1K
         NEXT          1K
         MINEXTENTS    1
         MAXEXTENTS  121
         PCTINCREASE   0
        )

/*
--------------------------------------------------------
FWSTAT_T.SQL
PURPOSE: SCRIPT USED TO CREATE FREEWAY TABLE - STATION
         FOR PRODUCTION.
        STATION INFORMATION TABLE.  PRIMARILY A SUPPORT
         LOOKUP TABLE.

CHANGE HISTORY
DATE        BY          PURPOSE
----------- ----------- -------------------------------
01-AUG-1996 G. GUNELSON INITIAL CREATION FOR PRODUCTION.
19-APR-1998 XINHONG TAN BENCHMARK TEST CREATION WITH CHANGE
--------------------------------------------------------
*/
CREATE TABLE station
 (station                              VARCHAR2(5)
 )
TABLESPACE fwglld
PCTFREE 10
STORAGE (INITIAL       1K
         NEXT          1K
         MINEXTENTS    1
         MAXEXTENTS  121
         PCTINCREASE   0
        )

/*
--------------------------------------------------------
FWSTRD_T.SQL
PURPOSE: SCRIPT USED TO CREATE FREEWAY TABLE - STATRDWY
```

```
        FOR PRODUCTION.
        STATION ROADWAY INTERSECTION TABLE FOR THE STATION
        AND ROADWAY TABLES.  MILE POINT IS INCLUDED.
CHANGE HISTORY
DATE        BY          PURPOSE
----------- ----------- ------------------------------
01-AUG-1996 G. GUNELSON INITIAL CREATION FOR PRODUCTION.
19-APR-1998 XINHONG TAN BENCHMARK TEST CREATION WITH CHANGE
--------------------------------------------------------
*/
CREATE TABLE statrdwy
 (station                                  NUMBER(5)   ,
  route                                    VARCHAR2(8) ,
  mp                                       NUMBER(7,3),
  dir                                      VARCHAR2(5),
  loctype                                  VARCHAR2(2)
 )
TABLESPACE fwgl1d
PCTFREE 10
STORAGE (INITIAL      1K
        NEXT          1K
        MINEXTENTS    1
        MAXEXTENTS  121
        PCTINCREASE   0
       )

/*
--------------------------------------------------------
FWTID_T.SQL
PURPOSE: SCRIPT USED TO CREATE FREEWAY TABLE - DETECTOR
        FOR PRODUCTION.
CHANGE HISTORY
DATE        BY          PURPOSE
----------- ----------- ------------------------------
10-MAR-1998 XINHONG TAN TEST CREATION FOR PRODUCTION.
--------------------------------------------------------
*/
CREATE TABLE datetime
 (timeid                    NUMBER(5)
  CONSTRAINT pk_timeid
  PRIMARY KEY
  USING INDEX TABLESPACE fwgl1i
      STORAGE (INITIAL      2K
              NEXT          2K
              MINEXTENTS    1
              MAXEXTENTS  121
              PCTINCREASE   0
              )                     ,
  readdate                  DATE      ,
  time                      VARCHAR2(4)),
  dayofweek                 CHAR(1)
TABLESPACE fwgl1d
PCTFREE 10
STORAGE (INITIAL      2K
        NEXT          2K
        MINEXTENTS    1
        MAXEXTENTS  121
        PCTINCREASE   0
       )
```

Sample Data
Fivemin Table

DETECTOR	TIMEID	VOL	OCC	VAL
1	0	0	0.0	2
2	0	0	0.0	1
3	0	0	0.0	1
4	0	0	0.0	1
5	0	0	0.0	1
6	0	0	0.0	1
7	0	0	0.0	1
8	0	17	1.0	0
9	0	13	1.0	0
10	0	6	0.0	1
11	0	4	0.0	0
12	0	1	0.0	1
13	0	1	0.0	1
14	0	21	2.0	0
15	0	13	2.0	0
16	0	0	99.0	1
17	0	14	1.0	0
18	0	24	2.0	0
19	0	8	0.0	0
20	0	1	0.0	1
21	0	12	0.0	0
22	0	14	0.0	0
23	0	10	0.0	0
24	0	0	100.0	1
25	0	0	100.0	1
26	0	17	1.0	0
27	0	14	0.0	0
28	0	0	0.0	1
29	0	0	0.0	1
30	0	11	0.0	0
31	0	6	0.0	0
32	0	20	1.0	0
33	0	11	0.0	0
34	0	4	0.0	1
35	0	1	0.0	1
36	0	5	0.0	0
37	0	0	100.0	1
38	0	6	0.0	0
39	0	14	0.0	0
40	0	8	0.0	0
41	0	14	1.0	0
42	0	0	0.0	1
43	0	13	0.0	0

Detector Table

DETECTOR	DETNAME	ZO	STATION
1	<Future>		
2	35W/CLIFFNMH		
3	35W/98THNMH		
4	35W/76THNMH		
5	35W/TH13NMH		
6	35W/66THNMH		
7	35W/98THSMH		
8	35W/TH13S1		80
9	35W/TH13S2		80
10	35W/TH13S3		80
11	35W/TH13SX		
12	35W/TH13SM		
13	35W/CR42NMH		
14	694/LONGW1		178
15	694/LONGW2		178
16	694/LONGW3		178
17	694/LONGE1		177
18	694/LONGE2		177
19	694/LONGE3		177
20	694/LONGWM		
21	94/PLYME1		
22	94/PLYME2		260
23	94/PLYME3		260
24	94/PLYME4		260
25	94/PLYME5		260
26	94/PLYMW1		
27	94/PLYMW2		259
28	94/PLYMW3		259
29	94/PLYMW4		259
30	94/PLYMW5		259
31	94/7STWM		
32	94/3STWML		
33	94/3STWMR		
34	94/4STEXL		
35	94/4STEXR		
36	94/7STEX		
37	94/BROADEX		
38	94/26AVE1		
39	94/26AVE2		
40	94/26AVE3		257
41	94/26AVE4		257
42	94/26AVE5		257
43	94/26AVW1		

Statrdwy Table

TATION	ROUTE	MP	DIR	LO
-1121	US212-D	161.439	SB	EN
-1324	US212-I	159.941	NB	EX
-1325	US212-I	160.104	NB	EN
-1328	US212-I	160.864	NB	EX
-1331	US212-I	161.196	NB	EN
-1031	US212-I	161.415	NB	EX
1	MN65-D	.921	SB	ML
2	I35W-D	16.252	SB	EN
2	MN65-D	.47	SB	ML
3	I35W-D	16.604	SB	ML
4	I35W-D	16.116	SB	ML
5	I35W-D	15.832	SB	ML
6	I35W-D	15.44	SB	ML
7	I35W-D	15.002	SB	ML
8	I35W-D	14.581	SB	ML
9	I35W-D	14.14	SB	ML
10	I35W-D	13.642	SB	ML
11	I35W-D	13.173	SB	ML
12	I35W-D	12.751	SB	ML
13	I35W-D	8.65	SB	ML
13	I35W-D	8.65	SB	HL
14	I35W-D	12.34	SB	ML
15	I35W-D	11.879	SB	ML
16	I35W-D	11.675	SB	ML
17	I35W-D	11.129	SB	ML
18	I35W-D	10.72	SB	ML
19	I35W-D	10.326	SB	ML
20	I35W-D	9.763	SB	ML
21	I35W-D	9.364	SB	ML
22	I35W-D	9.1	SB	ML
22	I35W-D	9.1	SB	HL
23	I35W-D	8.8	SB	ML
23	I35W-D	8.8	SB	HL
24	I35W-D	8.19	SB	ML
24	I35W-D	8.19	SB	HL
25	I35W-D	7.76	SB	ML
25	I35W-D	7.76	SB	HL
26	I35W-D	7.23	SB	ML
27	I35W-D	6.67	SB	ML
27	I35W-D	6.67	SB	HL
28	I35W-D	6.13	SB	ML
28	I35W-D	6.13	SB	HL
29	I35W-D	5.71	SB	ML

Xstreet Table

OUTE	XSTRNAME	MP
I94	Fish Lake Int	216.5
I94	Hemlock Ln	217.5
I94	TH 169	218.7
I94	Boone Ave	219.7
I94	Co Rd 81	220.5
I94	Zane Ave	221.3
I94	Brooklyn Blvd	222.4
I94	Xerxes Ave	223.2
I94	Shingle Creek Pkwy	223.6
I94	Humboldt Ave	224.2
I94	TH252	225
I94	57th Ave	225.5
I94	53rd Ave	225.7
I94	49th Ave	226.8
I94	42nd Ave	227.4
I94	Dowling Ave	227.7
I94	Lowry Ave	228.7
I94	26th Ave	229.2
I94	Broadway St	229.7
I94	TH55	230.6
I94	I394	231.2
I94	Tunnel East #1	231.8
I94	Tunnel East #2	231.9
I94	Tunnel East #3	232.1
I94	Tunnel West #1	232.1
I94	Tunnel West #2	231.9
I94	Tunnel West #3	231.8
I94	Hennepin Ave	232.2
I94	Lyndale Ave	232.2
I94	Groveland Ave	232.3
I94	LaSalle Ave	232.5
I94	3rd Ave	232.8
I94	TH65	233
I94	Portland Ave	233.1
I94	Park Ave	233.2
I94	11th Ave	233.6
I94	Cedar Ave	234.1
I94	Riverside Ave	234.8
I94	River Bridge	235.1
I94	Huron St	235.3
I94	Franklin Ave	235.8
I94	West of TH280	236.3
I94	TH280	236.5
I94	Vandalia St	236.9

Datetime Table

TIMEID	READDATE	TIME	D
0	01-OCT-97	0005	4
1	01-OCT-97	0010	4
2	01-OCT-97	0015	4
3	01-OCT-97	0020	4
4	01-OCT-97	0025	4
5	01-OCT-97	0030	4
6	01-OCT-97	0035	4
7	01-OCT-97	0040	4
8	01-OCT-97	0045	4
9	01-OCT-97	0050	4
10	01-OCT-97	0055	4
11	01-OCT-97	0100	4
12	01-OCT-97	0105	4
13	01-OCT-97	0110	4
14	01-OCT-97	0115	4
15	01-OCT-97	0120	4
16	01-OCT-97	0125	4
17	01-OCT-97	0130	4
18	01-OCT-97	0135	4
19	01-OCT-97	0140	4
20	01-OCT-97	0145	4
21	01-OCT-97	0150	4
22	01-OCT-97	0155	4
23	01-OCT-97	0200	4
24	01-OCT-97	0205	4
25	01-OCT-97	0210	4
26	01-OCT-97	0215	4
27	01-OCT-97	0220	4
28	01-OCT-97	0225	4
29	01-OCT-97	0230	4
30	01-OCT-97	0235	4
31	01-OCT-97	0240	4
32	01-OCT-97	0245	4
33	01-OCT-97	0250	4
34	01-OCT-97	0255	4
35	01-OCT-97	0300	4
36	01-OCT-97	0305	4
37	01-OCT-97	0310	4
38	01-OCT-97	0315	4
39	01-OCT-97	0320	4
40	01-OCT-97	0325	4
41	01-OCT-97	0330	4
42	01-OCT-97	0335	4
43	01-OCT-97	0340	4

APPENDIX D

Load Program

```
/*
--------------------------------------------------------
loaddet.ctl
PURPOSE:
LOAD DATA
INFILE 'H:\test\data\f_det.dat'
INSERT INTO table xtan.detector
(detector   POSITION (01:04)  INTEGER EXTERNAL,
detname     POSITION (10:23)  CHAR,
station     POSITION (24:29)  CHAR)


loadfmd.ctl
LOAD DATA
INFILE 'H:\test\data\19971023.dat'
APPEND INTO table xtan.fivemin
(detector   POSITION (01:06)  INTEGER EXTERNAL,
timeid      POSITION (07:14)  INTEGER EXTERNAL,
volume      POSITION (15:20)  INTEGER EXTERNAL,
occupancy   POSITION (21:26)  DECIMAL EXTERNAL,
validity    POSITION (28:28)  CHAR)

loadstatroute.ctl
LOAD DATA
INFILE 'H:\test\data\mp.dat'
INSERT INTO table xtan.statrdwy
(loctype    POSITION (01:02)  CHAR,
station     POSITION (09:14)  CHAR,
route       POSITION (17:24)  CHAR,
dir         POSITION (25:29)  CHAR,
mp          POSITION (34:40)  INTEGER EXTERNAL)

loadxstreet.ctl
LOAD DATA
INFILE 'H:\test\data\xstreet.dat'
INSERT INTO table xtan.statrdwy
(route       POSITION (1:7)  CHAR,
xstrname     POSITION (8:27)  CHAR,
mp           POSITION (30:37)  INTEGER EXTERNAL)
```

APPENDIX E

Conversion Program

```
#include <time.h>
#include <dos.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <iostream.h>
#include <ctype.h>
#include <math.h>

#define ERR_UNDEFINED 3
#define ERR_FUTURE 2
#define ERR_FLAGBAD 1
#define ERR_LINE -2
#define ERR_TIMEOUT -3
#define ERR_CHECKSUM -4
#define ERR_BAD -1
#define ERR_OK 0

// define the start date constant for one-period database
#define S_YEAR 1997
#define S_MONTH 10
#define S_DAY   1

// -------------
// 5 Min data file header
// -------------
typedef struct {
        char szSWLevel[8];
        char end1;
        char szFLLevel[8];
     char end2;
        unsigned short ZeroRes1;
        unsigned short ZeroRes2;
        unsigned short nDetectorCnt;
     unsigned short nStructureSize;
    unsigned short nDailyRecord;
    unsigned short tDataType;
    unsigned short dYear;
    unsigned char  dMonth;
    unsigned char  dDay;
} Data5MinFileHdr, *pData5MinFIleHdr;

// -----------------------------------------------
// Structure for 5-Minute detector data(binary) information
//
typedef struct {
        unsigned long  Vol: 8;          // 5-Minute Volume (0 to 255)
        unsigned long  Occ: 10;       // 5-Minute Occupancy (0.0% to 100.0%)
        unsigned long Status: 3;    // Communication status (-8 to +7)
        unsigned long Flag: 3;      // Detector flag code (-8 to +7)
} Int5MinData_t, *pInt5MinData_t;

// -----------------------------------------------
// Structure for 5-Minute detector data(ascii) information
//
typedef struct {
        long int Vol;
        float Occ;
        long int Status;
```

```
        unsigned short Flag;
        long TimeId;
} Xf5MinData_t, *Xpf5MinData_t;

// ------------------------------------------------
// Structure for 5-Minute detector timestamp
//
typedef struct {
        unsigned short                sHour;
        unsigned short                sMin;
        unsigned short                sSec;
} Time_t, *pTime_t;


// -----------------------------------
// Display pchMsg and exit the process
//
void ErrorOut(char *pchMsg)
{
        cout << pchMsg << "  " << endl;
        exit(3);
}

// ------------------------------------------------
// create timeid for each 5-minute interval
// each recordnum corresponds to an interval
//
int CreateTimeId(int dyear, int dmonth, int dday, Data5MinFileHdr &pData5MinFileHd
r,int RecordNum){
        int i;
        int daycnt;
        int days = 0;
        int timeid;
        int MonthDay[12] = {31,28,31,30,31,30,31,31,30,31,30,31};

        daycnt = pData5MinFileHdr.nDailyRecord;

        if(dyear != S_YEAR)
           return -1;
        if(dmonth != S_MONTH)
          {
           for(i = S_MONTH; i < dmonth; i++)
               days += MonthDay[i-1];
          }
        days = days + dday - S_DAY;
        timeid = days * daycnt + RecordNum;
        return timeid;
}

// -----------------------------------
// Display the header info of binary data file
//
void PrintHdr(Data5MinFileHdr &hdr)
{
        printf( "SZSWLevel: %s\n", hdr.szSWLevel);
                printf( "SZFLLevel: %s\n", hdr.szFLLevel);
        printf( "Detector Count: %6d\n", hdr.nDetectorCnt);
        printf( "Structure Size: %6d\n", hdr.nStructureSize);
        printf( "Daily Record:   %6d\n", hdr.nDailyRecord);
```

```
        printf( "Data Type:      %6d\n", hdr.tDataType);
}

// -----------------------------------
// Display the data info of binary data file
//
void PrintData(pInt5MinData_t &Det5MinData,
              int timeid, int detcnt)
{
       int i;

       printf("detector | volume  | occupancy | validity | timeid\n");
       for(i = 0; i < detcnt; i++)
          {
           printf("%10d%10d%10d%10.1f%10d\n",
                   i+1,
                                   timeid,
                   (int)Det5MinData[i].Vol,
                                   ((float)Det5MinData[i].Occ != 0.0 ? (float)Det5MinData[i].Occ / 10 : 0.0),
                   (int)Det5MinData[i].Flag);
          }
}

void PrintToFile(FILE *filename, pInt5MinData_t &Det5MinData,
                 int timeid, int detcnt)
{
        char cflag[1];
//      char cflag;

       for(int i = 0; i < detcnt; i++)
          {
                   _itoa((int)Det5MinData[i].Flag, cflag, 10);
                   //cflag = sflag[0];
           fprintf(filename, "%6d%8d%6d%6.1f%2s\n",
                   i+1,
                                   timeid,
                   (int)Det5MinData[i].Vol,
                   ((float)Det5MinData[i].Occ != 0.0 ? (float)Det5MinData[i].Occ / 10 : 0.0),
                   cflag);
          }
}


#include "tmc1.h"

main(int argc, char *argv[])
{
        FILE            *fp;                 //binary file pointer pointer
        FILE            *textFile;           //5 minute summary file pointer
        char            dataFile[30];        //name of input file (binary)
        char                outFileName[40];     //name of 5 minute file(ASCII)
        short           rc;                          //rc = record count
        Data5MinFileHdr hdr;//header for five minute file
        pInt5MinData_t  p5MinData; //record data
        short             i, j;                  //index counter
    long timeid;                         // time id
        int ryear;                       // data read year
        int rmonth;                      // data read year
        int rday;                        // data read year
```

```
char tmp1[4];                        // temporary char for string processing
    char tmp2[2];                    // temporary char for string processing
char indir[] = "Q:\\All Users\\Drexel\\Octdata\\";
    char outdir[] = "H:\\test\\data\\";
char pathname[40];


    //get input binary file name woithout .5mn suffix
    cout << "Reading traffic detector data from ";
    cin >> dataFile;
    cout << endl;

    //create output file names
    strcpy(outFileName, outdir);
    strcat(outFileName, dataFile);
strcat(outFileName, ".dat");
    cout << outFileName << endl;

    //get the year, month, day of date when data read
    strncpy(tmp1, dataFile,4);
    ryear = atoi(tmp1);

    tmp2[0] = dataFile[4];
    tmp2[1] = dataFile[5];
    rmonth = atoi(tmp2);

    tmp2[0] = dataFile[6];
    tmp2[1] = dataFile[7];
    rday = atoi(tmp2);

    //open output files
    if (!(textFile = fopen(outFileName,"wt")))
            ErrorOut("Error opening file out.txt\n");

//open binary file for reading
strcpy(pathname, indir);
    strcat(dataFile, ".5mn");
    strcat(pathname, dataFile);
    fp = fopen(pathname,"rb");

    if (fp == NULL)
    {
            perror("open binary");
            ErrorOut("Error opening data file");
    }

//read header information
    rc = fread(&hdr,sizeof(Data5MinFileHdr),1,fp);
    if (rc != 1)
       exit(2);
    else
       printf ("\n Size of struct: %d\n" , sizeof(Data5MinFileHdr));

    // print out the header file
PrintHdr(hdr);

    //close file
fclose(fp);
```

```
// open file again
    fp = fopen(pathname,"rb");

    if (fp == NULL)
    {
            perror("open binary");
            ErrorOut("Error opening data file");
    }


// read header
rc = fread(&hdr,sizeof(Data5MinFileHdr),1,fp);

    if (rc != 1)
       exit(2);
    // allocate space to the data record
    p5MinData = new Int5MinData_t;


    //read 5 minutes of data
    //for(i = 0; i < 2; i++)
for(i = 0; i < hdr.nDailyRecord; i++)
            {
            // calculate the time id for this time interval
            timeid = CreateTimeId(ryear, rmonth, rday,hdr, i);

            // in a loop of reading data record one by one
        for(j = 0; j < hdr.nDetectorCnt; j ++)
            //for(j = 0; j < 1000; j ++)
                    {
                 rc = fread(p5MinData,hdr.nStructureSize , 1, fp);
             if(rc != 1)
               ErrorOut("Error Reading data record");
                 else
                       PrintFile(textFile, p5MinData, timeid, j);
                    }
            }



free(p5MinData);

//close all the files
    fclose(textFile);
    fclose(fp);

    cout << endl << "Report Generation is Complete" << endl;

    return 0;

} //end main
```

APPENDIX F

CGI Script

```
#!D:\perl5\bin\perl.exe -w

use DBI;

print "Content-type: text/html\n\n";

%arg = &read_input;



if ( $arg{"action"} eq "Show SQL" ) {
    $n = "\n <BR>"; }
else { $n = ''; }

$tg = $arg{"time1"} ; # $time_granularity
$sg = $arg{"space1"} ; # $space_granularity
$lg = $arg{"sensor_select"} ; # $location specification type
$gg = $arg{"gazeteer_select"} ; # $gazeteer specification type
$dg = $arg{"sensor1"} ; # $sensor1 for location selection
$mg = $arg{"line_main"} ; # $line_main for location selection

$st = '-';

#initialize the variables

$dataTable = '';
$start_day = '';
$start_time = '';
$end_day = '';
$end_time = '';

#calculate some values to check for user input

$start_id = $arg{"start_hour"} * 12 + $arg{"start_minute"} / 5;
    $end_id = $arg{"end_hour"} * 12 + $arg{"end_minute"} / 5;

%month_num = (
    "JAN", 1,
    "FEB", 2,
    "MAR", 3,
    "APR", 4,
    "MAY", 5,
    "JUN", 6,
    "JUL", 7,
    "AUG", 8,
    "SEP", 9,
    "OCT", 10,
    "NOV", 11,
    "DEC", 12
);

%num_month = (
   1, "JAN",
   2, "FEB",
   3, "MAR",
   4, "APR",
   5, "MAY",
   6, "JUN",
   7, "JUL",
```

```
    8, "AUG",
    9, "SEP",
    10, "OCT",
    11, "NOV",
    12, "DEC"
);

%month_day = (
    1, 31,
    2, 28,
    3, 31,
    4, 30,
    5, 31,
    6, 30,
    7, 31,
    8, 31,
    9, 30,
    10, 31,
    11, 30,
    12, 31
);

$begin_in_year = 0;
$end_in_year = 0;

$begin_month = $month_num{$arg{"start_month"}};
$end_month = $month_num{$arg{"end_month"}};

for ($i = 1; $i <= $begin_month; $i++)
    {
     $begin_in_year += $month_day{$i};
    }
for ($i = 1; $i <= $end_month; $i++)
    {
     $end_in_year += $month_day{$i};
    }


if ( $arg{"action"} eq "Xstreet" )
   {
print<<First ;
<html>

<head>
  <meta http-equiv="Content-Type"
   content="text/html; charset=iso-8859-1">
  <title>Gui Design</title>
</head>

<body background="/Images/backgrnd.gif" bgcolor="#FFFFFF">

<table border="0" width="600">
  <tr>
    <td width="450"><center><h2>Extraction Interface for Metropolitan Traffic Data</h2></center></td>
  </tr>
</table>

<hr>
```

```
<A HREF="intro.html" SHAPE="rect" coords="10, 10, 140,63"> About</A>
 
<A HREF="example.html" SHAPE="rect" coords="10, 10, 140,63"> Examples</A>
 
<A HREF="misc.html" SHAPE="rect" coords="10, 10, 140,63"> Future Extensions</A>
 
<hr>

First
;

print<<Second ;
<form name="gui" action="http://ursa.itslab.umn.edu/cgi-bin/query_test.cgi" method="POST">
<table border="0">
          <tr>
             <td>   </td>
             <th> </th>
             <tr>Only for sensors on
              <input type="radio" checked name="line_main" value="all">All
              <input type="radio" name="line_main" value="mainline">Main Line
              <input type="radio" name="line_main" value="ramp">Ramp
               <select name="ramptype" ><option> Choose </option> <option>Exit</option>
               <option>Entrance</option><option>Both</option></select>
             </tr>
           </tr>
</table>
<table border="0">
          <tr><b>
   <font color=red>Use one of the two selection methods for specifying a set of sensors.</font></b><br>
  </tr>
  <tr>
             <td> </td>
             <td> </td>
             <td> </td>
              <td valign=top><input type="radio" checked name="sensor_select" value="gazeteer"><b>Sensor Locatio
              <td>
              <input type="radio"  checked name="gazeteer_select" value="route">Highway
              <Select name="highway" >
              <option> $arg{"highway"} </option>

              </select>

              Lane:<select name="lane">
              <option> $arg{"lane"} </option>
              </select>
              </td>
          </tr>
          <tr>
            <td> </td>
             <td> </td>
             <td> </td>
             <td> </td>
             <td align=left>
                 
                 
             <input type="radio"  checked name="startend_select" value="xstreet">Xstreet
              From<select name="street1"><option> Choose </option>

Second
```

```
;

if ( $arg{"highway"} eq "I94-EB" )
   {
    print<<I94EB ;

<option>Fish Lake Int(mp=216.5)</option>
<option>Hemlock Ln(mp=217.5)</option>
<option>TH 169(mp=218.7)</option>
<option>Boone Ave(mp=219.7)</option>
<option>Co Rd 81(mp=220.5)</option>
<option>Zane Ave(mp=221.3)</option>
<option>Brooklyn Blvd(mp=222.4)</option>
<option>Xerxes Ave(mp=223.2)</option>
<option>Shingle Creek Pkwy(mp=223.6)</option>
<option>Humboldt Ave(mp=224.2)</option>
<option>TH252(mp=225)</option>
<option>57th Ave(mp=225.5)</option>
<option>53rd Ave(mp=225.7)</option>
<option>49th Ave(mp=226.8)</option>
<option>42nd Ave(mp=227.4)</option>
<option>Dowling Ave(mp=227.7)</option>
<option>Lowry Ave(mp=228.7)</option>
<option>26th Ave(mp=229.2)</option>
<option>Broadway St(mp=229.7)</option>
<option>TH55(mp=230.6)</option>
<option>I394(mp=231.2)</option>
<option>Tunnel East #1(mp=231.8)</option>
<option>Tunnel West #3(mp=231.8)</option>
<option>Tunnel East #2(mp=231.9)</option>
<option>Tunnel West #2(mp=231.9)</option>
<option>Tunnel East #3(mp=232.1)</option>
<option>Tunnel West #1(mp=232.1)</option>
<option>Hennepin Ave(mp=232.2)</option>
<option>Lyndale Ave(mp=232.2)</option>
<option>Groveland Ave(mp=232.3)</option>
<option>LaSalle Ave(mp=232.5)</option>
<option>3rd Ave(mp=232.8)</option>
<option>TH65(mp=233)</option>
<option>Portland Ave(mp=233.1)</option>
<option>Park Ave(mp=233.2)</option>
<option>11th Ave(mp=233.6)</option>
<option>Cedar Ave(mp=234.1)</option>
<option>Riverside Ave(mp=234.8)</option>
<option>River Bridge(mp=235.1)</option>
<option>Huron St(mp=235.3)</option>
<option>Franklin Ave(mp=235.8)</option>
<option>West of TH280(mp=236.3)</option>
<option>TH280(mp=236.5)</option>
<option>Vandalia St(mp=236.9)</option>
<option>Cleveland Ave(mp=237.4)</option>
<option>Prior Ave(mp=237.7)</option>
<option>Fairview Ave(mp=237.9)</option>
<option>Snelling Ave(mp=238.5)</option>
<option>Hamline Ave(mp=238.9)</option>
<option>Lexington Ave(mp=239.6)</option>
<option>Victoria St(mp=239.9)</option>
<option>Dale St(mp=240.2)</option>
```

```
<option>Western Ave(mp=240.9)</option>
<option>Marion St(mp=241.2)</option>
<option>John Ireland Blvd(mp=241.4)</option>
<option>Wabasha St(mp=241.7)</option>
<option>Jackson St(mp=241.8)</option>
<option>I35E(mp=242.1)</option>
<option>East 7th St(mp=242.5)</option>
<option>TH52(mp=242.7)</option>
<option>Kellogg Blvd(mp=243.3)</option>
<option>Mounds Blvd(mp=243.5)</option>
<option>Earl St(mp=244.1)</option>
<option>TH61(mp=244.6)</option>
              </select>
              To<select name="street2">
<option>Fish Lake Int(mp=216.5)</option>
<option>Hemlock Ln(mp=217.5)</option>
<option>TH 169(mp=218.7)</option>
<option>Boone Ave(mp=219.7)</option>
<option>Co Rd 81(mp=220.5)</option>
<option>Zane Ave(mp=221.3)</option>
<option>Brooklyn Blvd(mp=222.4)</option>
<option>Xerxes Ave(mp=223.2)</option>
<option>Shingle Creek Pkwy(mp=223.6)</option>
<option>Humboldt Ave(mp=224.2)</option>
<option>TH252(mp=225)</option>
<option>57th Ave(mp=225.5)</option>
<option>53rd Ave(mp=225.7)</option>
<option>49th Ave(mp=226.8)</option>
<option>42nd Ave(mp=227.4)</option>
<option>Dowling Ave(mp=227.7)</option>
<option>Lowry Ave(mp=228.7)</option>
<option>26th Ave(mp=229.2)</option>
<option>Broadway St(mp=229.7)</option>
<option>TH55(mp=230.6)</option>
<option>I394(mp=231.2)</option>
<option>Tunnel East #1(mp=231.8)</option>
<option>Tunnel West #3(mp=231.8)</option>
<option>Tunnel East #2(mp=231.9)</option>
<option>Tunnel West #2(mp=231.9)</option>
<option>Tunnel East #3(mp=232.1)</option>
<option>Tunnel West #1(mp=232.1)</option>
<option>Hennepin Ave(mp=232.2)</option>
<option>Lyndale Ave(mp=232.2)</option>
<option>Groveland Ave(mp=232.3)</option>
<option>LaSalle Ave(mp=232.5)</option>
<option>3rd Ave(mp=232.8)</option>
<option>TH65(mp=233)</option>
<option>Portland Ave(mp=233.1)</option>
<option>Park Ave(mp=233.2)</option>
<option>11th Ave(mp=233.6)</option>
<option>Cedar Ave(mp=234.1)</option>
<option>Riverside Ave(mp=234.8)</option>
<option>River Bridge(mp=235.1)</option>
<option>Huron St(mp=235.3)</option>
<option>Franklin Ave(mp=235.8)</option>
<option>West of TH280(mp=236.3)</option>
<option>TH280(mp=236.5)</option>
<option>Vandalia St(mp=236.9)</option>
<option>Cleveland Ave(mp=237.4)</option>
```

```
<option>Prior Ave(mp=237.7)</option>
<option>Fairview Ave(mp=237.9)</option>
<option>Snelling Ave(mp=238.5)</option>
<option>Hamline Ave(mp=238.9)</option>
<option>Lexington Ave(mp=239.6)</option>
<option>Victoria St(mp=239.9)</option>
<option>Dale St(mp=240.2)</option>
<option>Western Ave(mp=240.9)</option>
<option>Marion St(mp=241.2)</option>
<option>John Ireland Blvd(mp=241.4)</option>
<option>Wabasha St(mp=241.7)</option>
<option>Jackson St(mp=241.8)</option>
<option>I35E(mp=242.1)</option>
<option>East 7th St(mp=242.5)</option>
<option>TH52(mp=242.7)</option>
<option>Kellogg Blvd(mp=243.3)</option>
<option>Mounds Blvd(mp=243.5)</option>
<option>Earl St(mp=244.1)</option>
<option>TH61(mp=244.6)</option>
              </select>
             </td>
          </tr>
I94EB
;
}

elsif ( $arg{"highway"} eq "I94-WB" )
   {
    print<<I94WB ;
<option>TH61(mp=244.6)</option>
<option>Earl St(mp=244.1)</option>
<option>Mounds Blvd(mp=243.5)</option>
<option>Kellogg Blvd(mp=243.3)</option>
<option>TH52(mp=242.7)</option>
<option>East 7th St(mp=242.5)</option>
<option>I35E(mp=242.1)</option>
<option>Jackson St(mp=241.8)</option>
<option>Wabasha St(mp=241.7)</option>
<option>John Ireland Blvd(mp=241.4)</option>
<option>Marion St(mp=241.2)</option>
<option>Western Ave(mp=240.9)</option>
<option>Dale St(mp=240.2)</option>
<option>Victoria St(mp=239.9)</option>
<option>Lexington Ave(mp=239.6)</option>
<option>Hamline Ave(mp=238.9)</option>
<option>Snelling Ave(mp=238.5)</option>
<option>Fairview Ave(mp=237.9)</option>
<option>Prior Ave(mp=237.7)</option>
<option>Cleveland Ave(mp=237.4)</option>
<option>Vandalia St(mp=236.9)</option>
<option>TH280(mp=236.5)</option>
<option>West of TH280(mp=236.3)</option>
<option>Franklin Ave(mp=235.8)</option>
<option>Huron St(mp=235.3)</option>
<option>River Bridge(mp=235.1)</option>
<option>Riverside Ave(mp=234.8)</option>
<option>Cedar Ave(mp=234.1)</option>
<option>11th Ave(mp=233.6)</option>
<option>Park Ave(mp=233.2)</option>
```

```
<option>Portland Ave(mp=233.1)</option>
<option>TH65(mp=233)</option>
<option>3rd Ave(mp=232.8)</option>
<option>LaSalle Ave(mp=232.5)</option>
<option>Groveland Ave(mp=232.3)</option>
<option>Hennepin Ave(mp=232.2)</option>
<option>Lyndale Ave(mp=232.2)</option>
<option>Tunnel East #3(mp=232.1)</option>
<option>Tunnel West #1(mp=232.1)</option>
<option>Tunnel East #2(mp=231.9)</option>
<option>Tunnel West #2(mp=231.9)</option>
<option>Tunnel East #1(mp=231.8)</option>
<option>Tunnel West #3(mp=231.8)</option>
<option>I394(mp=231.2)</option>
<option>TH55(mp=230.6)</option>
<option>Broadway St(mp=229.7)</option>
<option>26th Ave(mp=229.2)</option>
<option>Lowry Ave(mp=228.7)</option>
<option>Dowling Ave(mp=227.7)</option>
<option>42nd Ave(mp=227.4)</option>
<option>49th Ave(mp=226.8)</option>
<option>53rd Ave(mp=225.7)</option>
<option>57th Ave(mp=225.5)</option>
<option>TH252(mp=225)</option>
<option>Humboldt Ave(mp=224.2)</option>
<option>Shingle Creek Pkwy(mp=223.6)</option>
<option>Xerxes Ave(mp=223.2)</option>
<option>Brooklyn Blvd(mp=222.4)</option>
<option>Zane Ave(mp=221.3)</option>
<option>Co Rd 81(mp=220.5)</option>
<option>Boone Ave(mp=219.7)</option>
<option>TH 169(mp=218.7)</option>
<option>Hemlock Ln(mp=217.5)</option>
<option>Fish Lake Int(mp=216.5)</option>
</select>
To<select name="street2">
<option>TH61(mp=244.6)</option>
<option>Earl St(mp=244.1)</option>
<option>Mounds Blvd(mp=243.5)</option>
<option>Kellogg Blvd(mp=243.3)</option>
<option>TH52(mp=242.7)</option>
<option>East 7th St(mp=242.5)</option>
<option>I35E(mp=242.1)</option>
<option>Jackson St(mp=241.8)</option>
<option>Wabasha St(mp=241.7)</option>
<option>John Ireland Blvd(mp=241.4)</option>
<option>Marion St(mp=241.2)</option>
<option>Western Ave(mp=240.9)</option>
<option>Dale St(mp=240.2)</option>
<option>Victoria St(mp=239.9)</option>
<option>Lexington Ave(mp=239.6)</option>
<option>Hamline Ave(mp=238.9)</option>
<option>Snelling Ave(mp=238.5)</option>
<option>Fairview Ave(mp=237.9)</option>
<option>Prior Ave(mp=237.7)</option>
<option>Cleveland Ave(mp=237.4)</option>
<option>Vandalia St(mp=236.9)</option>
<option>TH280(mp=236.5)</option>
<option>West of TH280(mp=236.3)</option>
```

```
<option>Franklin Ave(mp=235.8)</option>
<option>Huron St(mp=235.3)</option>
<option>River Bridge(mp=235.1)</option>
<option>Riverside Ave(mp=234.8)</option>
<option>Cedar Ave(mp=234.1)</option>
<option>11th Ave(mp=233.6)</option>
<option>Park Ave(mp=233.2)</option>
<option>Portland Ave(mp=233.1)</option>
<option>TH65(mp=233)</option>
<option>3rd Ave(mp=232.8)</option>
<option>LaSalle Ave(mp=232.5)</option>
<option>Groveland Ave(mp=232.3)</option>
<option>Hennepin Ave(mp=232.2)</option>
<option>Lyndale Ave(mp=232.2)</option>
<option>Tunnel East #3(mp=232.1)</option>
<option>Tunnel West #1(mp=232.1)</option>
<option>Tunnel East #2(mp=231.9)</option>
<option>Tunnel West #2(mp=231.9)</option>
<option>Tunnel East #1(mp=231.8)</option>
<option>Tunnel West #3(mp=231.8)</option>
<option>I394(mp=231.2)</option>
<option>TH55(mp=230.6)</option>
<option>Broadway St(mp=229.7)</option>
<option>26th Ave(mp=229.2)</option>
<option>Lowry Ave(mp=228.7)</option>
<option>Dowling Ave(mp=227.7)</option>
<option>42nd Ave(mp=227.4)</option>
<option>49th Ave(mp=226.8)</option>
<option>53rd Ave(mp=225.7)</option>
<option>57th Ave(mp=225.5)</option>
<option>TH252(mp=225)</option>
<option>Humboldt Ave(mp=224.2)</option>
<option>Shingle Creek Pkwy(mp=223.6)</option>
<option>Xerxes Ave(mp=223.2)</option>
<option>Brooklyn Blvd(mp=222.4)</option>
<option>Zane Ave(mp=221.3)</option>
<option>Co Rd 81(mp=220.5)</option>
<option>Boone Ave(mp=219.7)</option>
<option>TH 169(mp=218.7)</option>
<option>Hemlock Ln(mp=217.5)</option>
<option>Fish Lake Int(mp=216.5)</option>
</select>
             </td>
          </tr>
I94WB
;
}

print<<Last ;
           <tr>
            <td> </td>
             <td> </td>
             <td> </td>
             <td> </td>
             <td align=left>
                 
                 
             <input type="radio"  name="startend_select" value="mp">
              Mile Point: From <input name="startmp" size=10>
```

```
            To <input name="endmp" size=10>
               
             <input type="submit" name="action" value="Show Detector,Station ID">
           </td>
         </tr>

         <tr>
           <td> </td>
           <td> </td>
           <td> </td>
           <td> </td>
           <td>
           <input type="radio" name="gazeteer_select" value="zone">Ramp Metering Zone
            <Select name="zone_list">
            <option>Zone List</option>
            <option>Not available</option></select>
           </td>
         </tr>


        <tr>
           <td> </td>
           <td> </td>
           <td> </td>
     <td><input type="radio" name="sensor_select" value="sensor"><b>Set of sensors</b></td>
     <td>
       <input type="radio" checked name="sensor1" value="detector">Detector(1-3250)
       <input type="radio" name="sensor1" value="station">Station(1-770)
               From:<input type="text" name="id1" size=4>
               To:<input type="text" name="id2" size=4>

               <input type="submit" name="action" value="Show Route,Milemark,Type">
             </td>
   </tr>
</table>
<hr>
<table border="0">
          <tr>
             <th><left><font color=red> Time Interval</font></left></th>
          </tr>
          <tr>
             <td>  <b>Start Date</b></td>
             <td>
              Year:<select name="start_year">
              <option> 1997 </option></select></td>
              <td>
              Month:<select name="start_month">
              <option> OCT </option>
             </select></td>
             <td>
              Day:<select name="start_day">
              <option> 00</option>
              <option> 01</option> <option> 02</option> <option> 03</option>
              <option> 04</option> <option> 05</option> <option> 06</option>
              <option> 07</option> <option> 08</option> <option> 09</option>
              <option> 10</option> <option> 11</option> <option> 12</option>
              <option> 13</option> <option> 14</option> <option> 15</option>
              <option> 16</option> <option> 17</option> <option> 18</option>
              <option> 19</option> <option> 20</option> <option> 21</option>
```

```
 <option> 22</option> <option> 23</option> </select></select>
</td>

<td><b>Time</b></td>
 <td>
 hour:<select name="start_hour">
 <option> 00</option> <option> 01</option> <option> 02</option>
 <option> 03</option> <option> 04</option> <option> 05</option>
 <option> 06</option> <option> 07</option> <option> 08</option>
 <option> 09</option> <option> 10</option> <option> 11</option>
 <option> 12</option> <option> 13</option> <option> 14</option>
 <option> 16</option> <option> 17</option> <option> 18</option>
 <option> 19</option> <option> 20</option> <option> 21</option>
 <option> 22</option> <option> 23</option> <option> 24</option></select>
  </td>
   <td>min:<select name="start_minute">
 <option> 05</option> <option> 10</option><option> 15</option>
 <option> 20</option> <option> 25</option> <option> 30</option>
 <option> 35</option> <option> 40</option> <option> 45</option>
 <option> 50</option> <option> 55</option><option> 00</option>
 </select>
</td>
<td>  </td>
</tr>
<tr>
<td>  <b>End Date</b></td>
<td>
 Year:<select name="end_year">
 <option> 1997 </option></select></td>
 <td>
 Month:<select name="end_month">
 <option> OCT </option>
 </select></td>
<td>
 Day:<select name="end_day">
 <option> 00</option>
 <option> 01</option> <option> 02</option> <option> 03</option>
 <option> 04</option> <option> 05</option> <option> 06</option>
 <option> 07</option> <option> 08</option> <option> 09</option>
 <option> 10</option> <option> 11</option> <option> 12</option>
 <option> 13</option> <option> 14</option> <option> 15</option>
 <option> 16</option> <option> 17</option> <option> 18</option>
 <option> 19</option> <option> 20</option> <option> 21</option>
 <option> 22</option> <option> 23</option> </select> </select>
</td>
<td><b>Time</b></td>
 <td>
 hour:<select name="end_hour">
 <option> 24</option> <option> 00</option><option> 01</option> <option> 02</opt
ion>
 <option> 03</option> <option> 04</option> <option> 05</option>
 <option> 06</option> <option> 07</option> <option> 08</option>
 <option> 09</option> <option> 10</option> <option> 11</option>
 <option> 12</option> <option> 13</option> <option> 14</option>
 <option> 16</option> <option> 17</option> <option> 18</option>
 <option> 19</option> <option> 20</option> <option> 21</option>
 <option> 22</option> <option> 23</option> </select>
  </td>
   <td>min:<select name="end_minute">
```

```
              <option> 00</option> <option> 05</option> <option> 10</option>
              <option> 15</option> <option> 20</option> <option> 25</option>
              <option> 30</option> <option> 35</option> <option> 40</option>
              <option> 45</option> <option> 50</option> <option> 55</option>
              </select>
             </td>
             <td>  </td>
             <td>
               <input type="submit" name="action" value="Show Calendar">
             </td>
           </tr>


</table>
<table border="0">
           <tr>
             <td>  </td>
             <td>  </td>
             <td>  </td>
             <td>
              <input type="checkbox" name="weekday" value="weekday">
              <b>Restrict to the following day(s) of the week</b>
              <input type="checkbox" name="d1" value="Mon">Mon
              <input type="checkbox" name="d2" value="Tues">Tue
              <input type="checkbox" name="d3" value="Wed">Wed
              <input type="checkbox" name="d4" value="Thu">Thu
              <input type="checkbox" name="d5" value="Fri">Fri
              <input type="checkbox" name="d6" value="Sat">Sat
              <input type="checkbox" name="d7" value="Sun">Sun
             </td>
           </tr>


</table>

<hr>

<table border="0">
           <tr>
     <th><font color=red>Granularity </font></th>
  </tr>
  <tr>
     <td> </td>
     <td valign=top><b>Space factor</b></td>
     <td>
     <input type="radio" checked name="space1" value="detector">Detector
             <input type="radio" name="space1" value="station">Station
             <input type="radio" name="space1" value="statdet">Station and Detector<br>
      </td>
   </tr>
 <tr>
     <td>
               
             </td>
    <td valign=top><b>Time Factor</b>
    </td>
    <td align=left>
      <input type="radio" name="time1" value="thirty">30s
              <input type="radio" checked name="time1" value="fmin">5 min
```

```
            <input type="radio" name="time1" value="aligned15min">Aligned 15min
            <input type="radio" name="time1" value="aligned1hr">Aligned 1hr
            <input type="radio" name="time1" value="daily">daily<br>
            </td>
          </tr>

</table>

<hr>

<table border="0">
          <tr>
    <left><th><font color=red>Attributes</font></th></left>
            <td>
              <input type="checkbox" checked name="attrib1" value="volume">Volume
            </td>
             <td> </td>
    <td>
         <input type="checkbox"  checked name="attrib2" value="occupancy">Occupancy
    </td>
             <td> </td>
    <td>
<input type="checkbox" name="attrib3" value="validity">Validity
     </td>
          </tr>
          <tr>
    <left><th valign=top><font color=red>Statistics</font></th></left>
            <td>
              <select name="Volume_Column">
              <option value="none">None
              <option value="min"> Minimum
              <option value="max">Maximum
              <option value="avg">Average
              <option value="sum">Sum
              </select>
            </td>
            <td>
               
            </td>
            <td>
      <select name="Occ_Column">
              <option value="none">None
              <option value="min"> Minimum
              <option value="max">Maximum
              <option value="avg">Average
              </select>
              <br>
            </td>
          </tr>

</table>

<hr>
  
  
 

<input type="submit" name="action" value="Submit as Table">
<input type="submit" name="action" value="Submit as Metrix">
```

```
<input type="submit" name="action" value="Show SQL">
<input type="reset" value="Clear">

<hr>
</form>
</body>
</html>
Last
;

}

elsif ( $arg{"action"} eq "Show Route,Milemark,Type" || $arg{"action"} eq "Show Detector,Station ID")
{

#initialize the array
  @tableAtoms = ();
  @attributeAtoms = ();
  @whereatoms = ();

#choose tables

  push(@tableAtoms, "xtan.detector") ;
  push(@tableAtoms, "xtan.statrdwy");
  $dataTable = "detector";

#choose attribute
  push(@attributeAtoms, "detector") ;
  push(@attributeAtoms, "statrdwy\.station") ;
  push(@attributeAtoms, "route") ;
  push(@attributeAtoms, "dir") ;
  push(@attributeAtoms, "mp") ;
  push(@attributeAtoms, "loctype") ;

#determine the WHERE clause
  if ( $lg eq "sensor" && $dg eq "detector" )
     {
      if ( $arg{'id2'} )
         {
          if ($arg{"id2"} > $arg{"id1"} )
             { push(@whereatoms,"$dataTable\.Detector BETWEEN '$arg{'id1'}' AND '$arg{'id2'}'"); }
          else { push(@whereatoms,"$dataTable\.Detector BETWEEN '$arg{'id2'}' AND '$arg{'id1'}'"); }
         }
      else
          {
           push(@whereatoms,"$dataTable\.Detector = '$arg{'id1'}'");
           }
     }
  elsif ( $lg eq "sensor" && $dg eq "station" )
       {
        if ( $arg{'id2'} )
           {
            if ($arg{"id2"} > $arg{"id1"} )
               { push(@whereatoms,"statrdwy\.station BETWEEN '$arg{'id1'}' AND '$arg{'id2'}'"); }
            else { push(@whereatoms,"statrdwy\.station BETWEEN '$arg{'id2'}' AND '$arg{'id1'}'"); }
           }
        else
           {
            push(@whereatoms,"statrdwy\.station = '$arg{'id1'}'");
```

```
        }
    }


if( $gg eq "route" && $lg eq "gazeteer")
  {

  @route = split(/-/, $arg{"highway"});
  $highway = $route[0];
  $direction = $route[1];

   if ( $direction eq "SB" || $direction eq "WB" )
     {
      push(@whereatoms, "statrdwy\.route = '$highway-D'");
     }
  else
    {
      push(@whereatoms, "statrdwy\.route = '$highway-I'");
    }

  if ( $arg{"startend_select"} eq "mp" )
     {
       if ( $arg{'startmp'} <= $arg{'endmp'} )
          {
           push(@whereatoms, "statrdwy\.mp \>= $arg{'startmp'}");
           push(@whereatoms, "statrdwy\.mp \<=  $arg{'endmp'}");
          }
       else
          {
           push(@whereatoms, "statrdwy\.mp \>= $arg{'endmp'}");
           push(@whereatoms, "statrdwy\.mp \<=  $arg{'startmp'}");
          }
     }
   else
     {

       @temp = split(/\(/, $arg{"street1"});
       $street1 = $temp[0];

       @temp = split(/\(/, $arg{"street2"});
       $street2 = $temp[0];

       $mp1 = "SELECT mp FROM xtan\.xstreet WHERE xstrname = '$street1' AND route = '$highway'";
       $mp2 = "SELECT mp FROM xtan\.xstreet WHERE xstrname = '$street2' AND route = '$highway'";

       $oradrh = DBI->install_driver( 'Oracle' );
       $dbh = DBI->connect( 'test', 'xtan', 'xinhong', 'Oracle' );
       $sth = $dbh->prepare( $mp1 );
       $sth->execute;
       while ( @row = $sth->fetchrow ) {
          $startmp = @row[0] ;       }
       $sth->finish;
       die unless $dbh;

       $sth = $dbh->prepare( $mp2 );
       $sth->execute;
       while ( @row = $sth->fetchrow ) {
          $endmp = @row[0] ;       }
       $sth->finish;
```

```
        die unless $dbh;
        $dbh->disconnect;

       if ($startmp <= $endmp )
          {
           push(@whereatoms, "statrdwy\.mp \>= $startmp");
            push(@whereatoms, "statrdwy\.mp \<=  $endmp");
           }
       else
           {
            push(@whereatoms, "statrdwy\.mp \>= $endmp");
            push(@whereatoms, "statrdwy\.mp \<=  $startmp");
           }
    }

  }

  push(@whereatoms,"detector.station = statrdwy.station");


  #prepare the SQL statement
  $attrs = join(", ", @attributeAtoms);
  $tables = join(", ", @tableAtoms);
  $conditions = join("$n AND ", @whereatoms );
  $sql = "SELECT unique $attrs $n FROM $tables $n WHERE $conditions$n " ;
 # print $sql;

#execute the sq;
  $num_attr = @attributeAtoms ;

  print "<CENTER>\n";
  print "<TABLE BORDER CELLPADDING=0 CELLSPACING=0 WIDTH=50%>\n";
  print "<CAPTION><H3>SENSOR LOACTION TABLE</H3></CAPTION>\n";
  print "<TR>\n";
  for($i = 0; $i < $num_attr; $i++)
        {
         if($attributeAtoms[$i] =~ /\./)
          {
           @subattr = split(/\./, $attributeAtoms[$i]);
           $num = @subattr;
           $attributeAtoms[$i] = $subattr[ $num - 1];
          }
         print "<TH>\n";
         print "$attributeAtoms[$i]" ;
         print "</TH>\n";
        }

  $oradrh = DBI->install_driver( 'Oracle' );
  $dbh = DBI->connect( 'test', 'xtan', 'xinhong', 'Oracle' );
  $sth = $dbh->prepare( $sql );
  $sth->execute;

   while ( @row = $sth->fetchrow )
         {
          print "<TR>\n";
          for($j = 0; $j < $num_attr; $j++)
            {
             print "<TD ALIGN=center>\n";
             print "$row[$j]" ;
```

```
            print "</TD>\n";
           }
          print "</TR>\n";
         }


     print "</TABLE>\n";
     print "</CENTER>\n";

     $sth->finish;

     die unless $dbh;

     $dbh->disconnect;

     exit;


}

elsif ( $arg{"action"} eq "Show Calendar" )
     {
      $space = ' ';
      @whereatoms = ();

      $sql = "SELECT unique to_char(readdate, 'DD'), dayofweek FROM datetime where";

      for ($i = $begin_month; $i <= $end_month; $i++)
         {
          $temp = $num_month{$i}.$st.$arg{"start_year"};
          $month_cond = "to_char(readdate, 'MON-YYYY') = '$temp'";
          $sql = $sql.$space.$month_cond;

          $oradrh = DBI->install_driver( 'Oracle' );
          $dbh = DBI->connect( 'test', 'xtan', 'xinhong', 'Oracle' );
          $sth = $dbh->prepare( $sql );
          $sth->execute;
          $days = 1;
          @mon_array = ();

          while ( @row = $sth->fetchrow )
               {
                if ($days == 1)
                   {
                    for ( $j = 1; $j < $row[1]; $j++ )
                        { push(@mon_array, $space); }
                   }
                $days++;
                push(@mon_array, $row[0]);

               }
          print "<CENTER>\n";
          print "<TABLE BORDER CELLPADDING=0 CELLSPACING=0 WIDTH=50%>\n";
          print "<CAPTION><H3>";
          print $num_month{$i};
          print $space;
          print $arg{"start_year"};
          print "</H3></CAPTION>\n";
          print "<TR><TH>Mon</TH><TH>Tues</TH><TH>Wed</TH><TH>Thur</TH><TH>Fri</TH><TH>Sat</TH><TH>Sun</TH></TR>'
```

```
        print "<TR>\n";
        $j = 1;

        foreach(@mon_array)
            {
             if(($j % 7) != 0)
               {
                print "<TD>\n";
                print "$_";
                print "</TD>\n";
               }
             else
               {
                print "<TD>\n";
                print "$_";
                print "</TD>\n";
                print "</TR>\n";
                print "<TR>\n";
               }
              $j++;
             }
        print "</TR>\n";
        print "</TABLE>\n";
        print "</CENTER>\n";

       }

}

elsif ( $arg{"action"} eq "Submit as Table"
|| $arg{"action"} eq "Submit as Metrix"
|| $arg{"action"} eq "Show SQL")
{
#-----------------------
#@whereatoms = list of atoms for where clause, e.g. "date > start date"
#@tableAtoms, e.g. Detector, DateTime, fivemin, fifteenmin, ohr,...
#@attributeAtoms, e.g. Detector.detector, fivemin.volume
#@groupbyAtoms, e.g. start_date, Detector.detector, ...
#After creating the arrays, we join each array into a string
# using proper separator, e.g. AND, comma etc.
# These string concatenated together create the SQL query

#--Table list--------------------
 # minimal set = DateTime, one of (fivemin, fifteenmin, onehr, daily)
 # Optional set =  Detector
 # Future set w/ zone gazeteer =  PlanZone, PlanRoute, Route, DetRoute
 # Future set w/ 30 second =  30sec table

@tableAtoms = () ;


 if ( $tg eq "fmin" )
   {

    $dataTable = "xtan.p_fivemin" ;
    push(@tableAtoms, "xtan.p_fivemin") ;
    push(@tableAtoms, "xtan.datetime") ;

    if ( $sg eq "station" || $sg eq "statdet" )
```

```
        {
          $dataTable_s = "xtan.v_stat_five" ;
          push(@tableAtoms_s, "xtan.v_stat_five") ;
          if ($gg ne "zone")
            { push(@tableAtoms, "xtan.detector") ; }

        }

    }

 elsif ( $tg eq "aligned15min" )
    {
     $dataTable = "xtan.v_ftmin" ;
     push(@tableAtoms, "xtan.v_ftmin") ;

     if ( $sg eq "station" || $sg eq "statdet" )
        { $dataTable_s = "xtan.v_stat_ftmin" ;
          push(@tableAtoms_s, "xtan.v_stat_ftmin") ;
          if ($gg ne "zone")
            { push(@tableAtoms, "xtan.detector") ; }
        }
    }
 elsif ( $tg eq "aligned1hr" )
    {
     $dataTable = "xtan.v_hour" ;
     push(@tableAtoms, "xtan.v_hour") ;
     if ( $sg eq "station" || $sg eq "statdet" )
        {
         $dataTable_s = "xtan.v_stat_hour" ;
         push(@tableAtoms_s, "xtan.v_stat_hour") ;
         if ($gg ne "zone")
            { push(@tableAtoms, "xtan.detector") ; }
        }
    }
 elsif ( $tg eq "daily" )
    {
        $dataTable = "xtan.v_daily" ;
        push(@tableAtoms, "xtan.v_daily") ;
        if ( $sg eq "station" || $sg eq "statdet" )
           {
            $dataTable_s = "xtan.v_stat_daily" ;
            push(@tableAtoms_s, "xtan.v_stat_daily") ;
            if ($gg ne "zone")
               { push(@tableAtoms, "xtan.detector") ; }
           }
    }
 elsif ($tg) {print "ERROR: Unknown value of time_granularity : $tg $n" ;
      print "Do not have minimal set of tables!!! Invalid SQL query$n";
}

 if ($sg eq "detector")
    {
     if ( $gg ne "zone" && ($lg ne "sensor" || $dg ne "detector") )
        {
          push(@tableAtoms, "xtan.detector");
        }
    }
 if ( $gg eq "route" && $lg eq "gazeteer")
    {
```

```
     push(@tableAtoms, "xtan.statrdwy");
     if ( $sg eq "station" || $sg eq "statdet" )
        {
         push(@tableAtoms_s, "xtan.statrdwy");
        }
    }


 if ($lg eq "sensor" && $mg ne "all")
     {
       if ( $dg eq "detector" )
           { push(@tableAtoms, "xtan.detector"); }
       push(@tableAtoms, "xtan.statrdwy");
       if ( $sg eq "station" || $sg eq "statdet" )
        {
         push(@tableAtoms_s, "xtan.statrdwy");
        }
      }

#--Attribute list--------------------

 # assume $tg contains time_granularity from table selection!!!
 # and $dataTable contains a table name from
 # fivemin, fifteenmin, onehr, daily, in future 30sec.

 # minimal set = ReadDate, Time
 # Optional = Volume, Occupancy, Validity, Detector, Station, Zone
 # Future option w/ zone gazetter = PlanRoute, DetRoute, Route

 @attributeAtoms = ();
 @attributeAtoms_s = ();
 $stat1 = $arg{"Volume_Column"} ;
 $stat2 = $arg{"Occ_Column"} ;
 if ( $stat1 eq "none" && $stat2 eq "none" )
    {
     push(@attributeAtoms, "readdate");
    }
 if ( $tg eq "fmin" || $tg eq "aligned15min") { push(@attributeAtoms, "time"); }
 elsif ( $tg eq "aligned1hr" )   { push(@attributeAtoms, "hour"); }

 if ($sg eq "station" || $sg eq "statdet" )
    {
     @attributeAtoms_s = @attributeAtoms;
    }
#choosing detector, station, zone attributes

 if ($sg eq "station" || $sg eq "statdet")
    {
     push(@attributeAtoms_s, "$dataTable_s\.station");
     if ($gg ne "zone")
        { push(@attributeAtoms, "xtan\.detector\.station") ; }
     else
        { push(@attributeAtoms, "xtan\.testzone\.station") ; }
    }
 push(@attributeAtoms, "$dataTable\.detector") ;

 #choosing volume, occupancy, validity attributes
$attr_count = 0;
```

```
if (($stat2 ne "none")&&($stat2 ne "max")
&&($stat2 ne "min")&&($stat2 ne "avg") )
{  print 'ERROR: Illegal $arg{"Occ_Column"} = ', "$stat2$n"; }

 if ( ($arg{"attrib2"}) && ($stat2 ne "none") )
     {
      $attr_count += 1;
      if ($sg eq "station" || $sg eq "statdet" )
        { push(@attributeAtoms_s, "$stat2(occupancy)") ; }
      push(@attributeAtoms, "$stat2(occupancy)") ;
     }
 elsif ( $arg{"attrib2"} )
      {
       $attr_count += 1;
       if ($sg eq "station" || $sg eq "statdet" )
         { push(@attributeAtoms_s, 'occupancy') ; }
       push(@attributeAtoms, 'occupancy') ;
      }

if (($stat2 ne "none")&&($stat2 ne "max")
&&($stat2 ne "min")&&($stat2 ne "avg") )
{  print 'ERROR: Illegal $arg{"Occ_Column"} = ', "$stat2$n"; }

 if ( ($arg{"attrib1"}) && ($stat1 ne "none") )
    {
      $attr_count += 1;
      if ($sg eq "station" || $sg eq "statdet" )
        { push(@attributeAtoms_s, "$stat1(volume)") ; }
      push(@attributeAtoms, "$stat1(volume)") ;
    }
 elsif ( $arg{"attrib1"} )
       {
        $attr_count += 1;
        if ($sg eq "station" || $sg eq "statdet" )
            { push(@attributeAtoms_s, 'volume') ; }
        push(@attributeAtoms, 'volume') ;
       }



 if ( $arg{"attrib3"} )
    {
     $attr_count += 1;
     if ($sg eq "station" || $sg eq "statdet" )
        { push(@attributeAtoms_s, 'validity') ; }
     push(@attributeAtoms, 'validity') ;
    }


#--Where clauses--in-@whereatoms-----------------
 # Minimal = join condition, date restriction
 # Maximal : restrict start_date, end_date, sensor_list
 # Future: allow restriction on volume, occupany, validity,
 # and gazeteer and map, mainline/ramp locations

@whereatoms = ();
if ($sg eq "station" || $sg eq "statdet" )
```

```
    {
     @whereatoms_s = ();
    }
$days = 0;
#form $start_date, $start_time, $end_date, $end_time
$st = '-';
$startyear = substr($arg{"start_year"},2,2);
$endyear = substr($arg{"end_year"},2,2);

if (  $arg{"start_day"} eq "00" )
   {
    $start_date = $arg{"start_month"}.$st.$arg{"start_year"} ;
    $days = 0;
    push(@whereatoms, "to_char(readdate, 'MON-RR') = '$start_date'");
   }
elsif ( $arg{"end_year"} eq "96" )
      {
       $start_date = $arg{"start_day"}.$st.$arg{"start_month"}.$st.$arg{"start_year"} ;
       push(@whereatoms, "ReadDate = to_date('$start_date', 'DD-MON-YYYY')");
      }
else
   {
    $start_date = $arg{"start_day"}.$st.$arg{"start_month"}.$st.$arg{"start_year"} ;
    $end_date = $arg{"end_day"}.$st.$arg{"end_month"}.$st.$arg{"end_year"} ;
    if ( $start_date eq $end_date )
      {push(@whereatoms, "ReadDate = to_date('$start_date', 'DD-MON-YYYY')");}
    else
      {
       $days = 1;
       if( $start_in_year < $end_in_year)
          {  push(@whereatoms, "ReadDate BETWEEN to_date('$start_date', 'DD-MON-YYYY') AND to_date('$end_date',
       else {   push(@whereatoms, "ReadDate BETWEEN to_date('$end_date', 'DD-MON-YYYY') AND to_date('$start_dat
      }
   }

if (! $arg{"start_hour"}) { $arg{"start_hour"} = "00" ; }
if (! $arg{"start_minute"}) { $arg{"start_minute"} = "05" ; }
if ( $arg{"end_hour"} eq "00") { $arg{"end_hour"} = "24" ; }
if ( $arg{"end_minute"} eq "00") { $arg{"end_minute"} = "00" ; }
if ( $tg eq "aligned1hr")
   {
   $start_time = $arg{"start_hour"} + 1;
   $end_time = $arg{"end_hour"};
    if ($startid < $end_id)
        { push(@whereatoms, "hour BETWEEN '$start_time' AND '$end_time'"); }
    else { push(@whereatoms, "hour BETWEEN '$end_time' AND '$start_time'"); }
   }
else
   {
    $start_time = $arg{"start_hour"} . $arg{"start_minute"} ;
    if ( $arg{"end_hour"} eq "00") { $arg{"end_hour"} = "24" ; }
    if ( $arg{"end_minute"} eq "00") { $arg{"end_minute"} = "00" ; }
    $end_time = $arg{"end_hour"} . $arg{"end_minute"} ;
    if( $tg ne "daily" )
       {
        if ($startid < $end_id)
           { push(@whereatoms, "time BETWEEN '$start_time' AND '$end_time'"); }
        else { push(@whereatoms, "time BETWEEN '$end_time' AND '$start_time'"); }
       }
```

```
   }
#form a list of the day of the week @dayList
#IN operator - check if strings are allowed

$ct = 0;
@dayList = () ;
if ( $arg{"weekday"} eq "weekday" )
   {
    if ($arg{"d1"} eq "Mon"){$ct = $ct + 1; push(@dayList, "'2'") ;  }
    if ($arg{"d2"} eq "Tues"){$ct = $ct + 1; push(@dayList, "'3'") ;  }
    if ($arg{"d3"} eq "Wed"){$ct = $ct + 1; push(@dayList, "'4'") ;  }
    if ($arg{"d4"} eq "Thu"){$ct = $ct + 1; push(@dayList, "'5'") ;  }
    if ($arg{"d5"} eq "Fri"){$ct = $ct + 1; push(@dayList, "'6'") ;  }
    if ($arg{"d6"} eq "Sat"){$ct = $ct + 1; push(@dayList, "'7'") ;  }
    if ($arg{"d7"} eq "Sun"){$ct = $ct + 1; push(@dayList, "'1'") ;  }
   }
if ($ct != 7)
  {
   $daylist = join(",", @dayList);
   if (@dayList)
     { push(@whereatoms, "DayofWeek IN ($daylist)");}
  }

if ($sg eq "station" || $sg eq "statdet" )
    {
     @whereatoms_s = @whereatoms ;
    }


if ( $lg eq "sensor" && $dg eq "detector" )
{
  if ( $arg{'id2'} )
     {
      if ($arg{"id2"} > $arg{"id1"} )
          { push(@whereatoms,"$dataTable\.Detector BETWEEN '$arg{'id1'}' AND '$arg{'id2'}'"); }
      else { push(@whereatoms,"$dataTable\.Detector BETWEEN '$arg{'id2'}' AND '$arg{'id1'}'"); }
     }
  else
     {
      push(@whereatoms,"$dataTable\.Detector = '$arg{'id1'}'");
     }
}
elsif ( $lg eq "sensor" && $dg eq "station" )
{

      if ( $arg{'id2'} )
         {
          if ($arg{"id2"} > $arg{"id1"} )
             { push(@whereatoms,"detector\.station BETWEEN '$arg{'id1'}' AND '$arg{'id2'}'"); }
          else { push(@whereatoms,"detector\.station BETWEEN '$arg{'id2'}' AND '$arg{'id1'}'"); }

         }
      else
         {
          push(@whereatoms,"detector\.station = '$arg{'id1'}'");
         }

      if ( $sg eq "station" || $sg eq "statdet")
         {
```

```
          if ( $arg{'id2'} )
             {
              if ($arg{"id2"} > $arg{"id1"} )
                  { push(@whereatoms_s,"$dataTable_s\.station BETWEEN '$arg{'id1'}' AND '$arg{'id2'}'"); }
              else { push(@whereatoms_s,"$dataTable_s\.station BETWEEN '$arg{'id2'}' AND '$arg{'id1'}'"); }
             }
          else
             {
             push(@whereatoms_s,"$dataTable_s\.station = '$arg{'id1'}'");
             }
        }
}

if( $gg eq "route" && $lg eq "gazeteer")
  {

    @route = split(/-/, $arg{"highway"});
    $highway = $route[0];
    $direction = $route[1];

     if ( $direction eq "SB" || $direction eq "WB" )
       {
        if ( $sg eq "station" || $sg eq "statdet")
           { push(@whereatoms_s, "statrdwy\.route = '$highway-D'"); }
        push(@whereatoms, "statrdwy\.route = '$highway-D'");
       }
    else
      {
        if ( $sg eq "station" || $sg eq "statdet")
          { push(@whereatoms_s, "statrdwy\.route = '$highway-I'");}

        push(@whereatoms, "statrdwy\.route = '$highway-I'");
      }

    if ( $arg{"startend_select"} eq "mp" )
       {
         if ( $arg{'startmp'} <= $arg{'endmp'} )
            {
             if ( $sg eq "station" || $sg eq "statdet")
              {
                push(@whereatoms_s, "statrdwy\.mp \>= $arg{'startmp'}");
                push(@whereatoms_s, "statrdwy\.mp \<=  $arg{'endmp'}");
              }
             push(@whereatoms, "statrdwy\.mp \>= $arg{'startmp'}");
             push(@whereatoms, "statrdwy\.mp \<=  $arg{'endmp'}");
            }
         else
            {
              if ( $sg eq "station" || $sg eq "statdet")
                 {
                  push(@whereatoms_s, "statrdwy\.mp \>= $arg{'endmp'}");
                  push(@whereatoms_s, "statrdwy\.mp \<=  $arg{'startmp'}");
                 }
             push(@whereatoms, "statrdwy\.mp \>= $arg{'endmp'}");
             push(@whereatoms, "statrdwy\.mp \<=  $arg{'startmp'}");
            }
       }
    else
       {
```

```
        @temp = split(/\(/, $arg{"street1"});
        $street1 = $temp[0];

        @temp = split(/\(/, $arg{"street2"});
        $street2 = $temp[0];

        $mp1 = "SELECT mp FROM xtan\.xstreet WHERE xstrname = '$street1' AND route = '$highway'";
        $mp2 = "SELECT mp FROM xtan\.xstreet WHERE xstrname = '$street2' AND route = '$highway'";

        $oradrh = DBI->install_driver( 'Oracle' );
        $dbh = DBI->connect( 'test', 'xtan', 'xinhong', 'Oracle' );
        $sth = $dbh->prepare( $mp1 );
        $sth->execute;
        while ( @row = $sth->fetchrow ) {
           $startmp = @row[0] ;       }
        $sth->finish;
        die unless $dbh;

        $sth = $dbh->prepare( $mp2 );
        $sth->execute;
        while ( @row = $sth->fetchrow ) {
           $endmp = @row[0] ;       }
        $sth->finish;
        die unless $dbh;
        $dbh->disconnect;

       if ($startmp <= $endmp )
          {
           if ( $sg eq "station" || $sg eq "statdet")
              {
               push(@whereatoms_s, "statrdwy\.mp \>= $startmp");
               push(@whereatoms_s, "statrdwy\.mp \<=  $endmp");
              }
            push(@whereatoms, "statrdwy\.mp \>= $startmp");
            push(@whereatoms, "statrdwy\.mp \<=  $endmp");
           }
       else
           {
            if ( $sg eq "station" || $sg eq "statdet")
               {
                push(@whereatoms_s, "statrdwy\.mp \>= $endmp");
                push(@whereatoms_s, "statrdwy\.mp \<=  $startmp");
               }
            push(@whereatoms, "statrdwy\.mp \>= $endmp");
            push(@whereatoms, "statrdwy\.mp \<=  $startmp");
           }
    }

  }
if( $gg eq "zone" && $lg eq "gazeteer")
  {
   push(@whereatoms, "detector\.zone = '$arg{'zone_list'}'");
   if ( $sg eq "station" || $sg eq "statdet")
      {
       push(@whereatoms_s, "detector\.zone = '$arg{'zone_list'}'");
      }
   }
```

```
if ( $mg ne "all" )
  {
   if ( $mg eq "mainline" )
       {
            push(@whereatoms, "statrdwy\.loctype = 'ML'");
            if ( $sg eq "station" || $sg eq "statdet")
               {
                push(@whereatoms_s, "statrdwy\.loctype = 'ML'");
               }
       }
   elsif ( $mg eq "ramp" )
         {
                 if ( $arg{"ramptype"} eq "Both" ) { $union = 1; }
                 if ( $arg{"ramptype"} eq "Exit" || $arg{"ramptype"} eq "Both" )
                     {
                      push(@whereatoms, "statrdwy\.loctype = 'EX'");
                      if ( $sg eq "station" || $sg eq "statdet")
                         {
                          push(@whereatoms_s, "statrdwy\.loctype = 'EX'");
                         }
                     }
                  elsif ( $arg{"ramptype"} eq "Entrance" )
                     {
                      push(@whereatoms, "statrdwy\.loctype = 'EN'");
                      if ( $sg eq "station" || $sg eq "statdet")
                         {
                          push(@whereatoms_s, "statrdwy\.loctype = 'EN'");
                         }
                     }
         }
   }


# join
   if ( $tg eq "fmin" )
     {
       push(@whereatoms,"$dataTable\.timeid = datetime.timeid");
     }
   if ( $lg eq "sensor" && $dg eq "station" )
     {
       push(@whereatoms,"$dataTable\.detector = detector.detector");
     }
   if ( $lg eq "sensor" && $mg ne "all" )
      {
        if ( $dg eq "detector" )
           {
             push(@whereatoms,"$dataTable\.detector= detector.detector");
             push(@whereatoms,"detector.station = statrdwy.station");
           }
        else
           { push(@whereatoms,"detector\.station = statrdwy.station"); }
      }
   if ( $lg eq "gazeteer" && $gg eq "route" )
     {
         push(@whereatoms,"$dataTable\.detector = detector.detector");
         push(@whereatoms,"detector.station = statrdwy.station");
     }
```

```
   if ( $sg eq "station" || $sg eq "statdet" )
    {
      if ( $lg eq "sensor" && $mg ne "all" )
      {
        if ( $dg eq "detector" )
           {
             push(@whereatoms_s,"$dataTable_s\.detector= detector.detector");
             push(@whereatoms_s,"detector.station = statrdwy.station");
           }
        else
              { push(@whereatoms_s,"$dataTable_s\.station = statrdwy.station"); }
       }
      if (  $gg eq "route" && $lg eq "gazeteer")
         {
          push(@whereatoms_s,"$dataTable_s\.station = statrdwy.station");
         }
      if ( $lg eq "gazeteer" && $gg eq "zone" )
         {
          push(@whereatoms_s,"$dataTable_s\.station = detector.station");
         }
    }


#filter
#$arg{"line_main"}
#$arg{"space1"} #for granularity of data - NEED IN FUTURE

#---gazeteer----affects attribute list and where clause
#--Group By list--------------------
# Assume variable $start_date , $start_time set by where clause processing
# and $$dataTable set by tablelist processing

# minimal set = ()
# Final set has to be a subset of the list of attributes in @attributeAtoms
# i.e.   ReadDate, Time, Detector, Station, Zone
# where ReadDate parts (e.g. Year, Month, Day of the Week) may be used!
# Future option w/ zone gazetter = PlanRoute, DetRoute, Route

@groupbyAtoms = () ;
if ( $sg eq "station" || $sg eq "statdet" )
   {
    @groupbyAtoms_s = () ;
   }
if ($arg{"Volume_Column"} ne "none" || $arg{"Occ_Column"} ne "none")
  {

        push(@groupbyAtoms, "$dataTable\.detector" );

        if ( $sg eq "station" || $sg eq "statdet" )
           {
            push(@groupbyAtoms_s, "$dataTable_s\.station" );
           }

        if ( $tg eq "aligned1hr" )
          {
           push(@groupbyAtoms, "hour" );
           if ( $sg eq "station" || $sg eq "statdet" )
```

```
            { push(@groupbyAtoms_s, "hour" ); }
         }
       if ( $tg ne "aligned1hr" )
         {
          push(@groupbyAtoms, "time" );
          if ( $sg eq "station" || $sg eq "statdet" )
             { push(@groupbyAtoms_s, "time" ); }
        }
  }

#------------Create SQL statement---------

$num_attr = @attributeAtoms ;
if ( $#tableAtoms > 0 )
   { $select = "SELECT /*+ STAR */"; }
else { $select = "SELECT"; }

$attrs = join(", ", @attributeAtoms);
$tables = join(", ", @tableAtoms);
$conditions = join("$n AND ", @whereatoms );
$sql = "$select $attrs $n FROM $tables $n WHERE $conditions$n " ;

if ( $sg eq "station" || $sg eq "statdet" )
   {
    $attrs_s = join(", ", @attributeAtoms_s);
    $tables_s = join(", ", @tableAtoms_s);
    $conditions_s = join("$n AND ", @whereatoms_s );
    if ( $gg eq "zone" && $lg eq "gazeteer")
       { $sql_s = "$select unique $attrs_s $n FROM $tables_s $n WHERE $conditions_s$n " ; }
    else { $sql_s = "$select $attrs_s $n FROM $tables_s $n WHERE $conditions_s$n " ; }
   }

if ( @groupbyAtoms ) {
$groupby = join(", ", @groupbyAtoms);
        $sql = $sql . "GROUP BY $groupby$n " ;
}

if($arg{"ramptype"} eq "Both" )
  {
   $temp = $sql;
   $temp =~ s/EX/EN/g;
   $sql = "$sql UNION $n $temp";
  }

if ( $sg eq "station" || $sg eq "statdet" )
   {
       if ( @groupbyAtoms_s )
          {
   $groupby_s = join(", ", @groupbyAtoms_s);
           $sql_s = $sql_s . "GROUP BY $groupby_s$n " ;
          }
       if ($arg{"ramptype"} eq "Both" )
          {
           $temp = $sql_s;
           $temp =~ s/EX/EN/g;
           $sql_s = "$sql_s UNION $n $temp";
          }
    }
```

```
#print $sql;

#if matrix format output needed, calculate the days and
# number of interval per day first

if ( $arg{"action"} eq "Submit as Metrix" )
   {
    $start_end = $end_id - $start_id + 1;

    $days = 1;

    if ( $arg{"start_month"} eq $arg{"end_month"} )
       {
        $days+=($arg{"end_day"} - $arg{"start_day"});
       }
    else
      {
       $start_month = $month_num{$arg{"start_month"}};
       $num_month = $month_num{$arg{"end_month"}} - $month_num{$arg{"start_month"}} + 1;

       foreach $i(1..$num_month)
           {
            if ($i == 1)
               {
                $days+=($month_day{$start_month} - $arg{"start_day"});
               }
            elsif ( $i == $num_month)
                  {
                   $days+=$arg{"end_day"};
                  }
            else
               {
                $month = $start_month + $i;
                $days+=$month_day{$month};
               }
           }
       }

     $num_interval = $days * $start_end;
  }

$num_attr_s = @attributeAtoms_s ;
$num_attr = @attributeAtoms ;
$non_attr_num_s = $num_attr_s - $attr_count;
$non_attr_num = $num_attr - $attr_count;
if ($sg ne "detector")
   { $time_num = $non_attr_num_s - 1; }
else
   { $time_num = $non_attr_num - 1; }
# print "here";

#make the attribute as reuired form
if ( $sg ne "station" )
   {
     for($i = 0; $i < $num_attr; $i++)
        {
         if($attributeAtoms[$i] =~ /\./)
```

```
          {
           @subattr = split(/\./, $attributeAtoms[$i]);
           $num = @subattr;
           $attributeAtoms[$i] = $subattr[ $num - 1];
          }
          if ( $attributeAtoms[$i] eq "occupancy" )
             {  $attributeAtoms[$i] = "Occ"; }
          elsif ( $attributeAtoms[$i] eq "volume" )
             {  $attributeAtoms[$i] = "Vol"; }
          elsif ( $attributeAtoms[$i] eq "validity" )
             {  $attributeAtoms[$i] = "Val"; }
          elsif ( $attributeAtoms[$i] eq "readdate" )
             {  $attributeAtoms[$i] = "Date"; }
        }
   }
if ($sg ne "detector" )
   {
    for($i = 0; $i < $num_attr_s; $i++)
        {
         if($attributeAtoms_s[$i] =~ /\./)
          {
           @subattr = split(/\./, $attributeAtoms_s[$i]);
           $num = @subattr;
           $attributeAtoms_s[$i] = $subattr[ $num - 1];
          }
          if ( $attributeAtoms_s[$i] eq "occupancy" )
             {  $attributeAtoms_s[$i] = "Occ"; }
          elsif ( $attributeAtoms_s[$i] eq "volume" )
             {  $attributeAtoms_s[$i] = "Vol"; }
          elsif ( $attributeAtoms_s[$i] eq "validity" )
             {  $attributeAtoms_s[$i] = "Val"; }
          elsif ( $attributeAtoms_s[$i] eq "readdate" )
             {  $attributeAtoms_s[$i] = "Date"; }
        }
    }


if ( $arg{"action"} eq "Submit as Table" )
 {
  if ( $sg eq "detector" )
     {
      print "<CENTER>\n";
      print "<TABLE BORDER CELLPADDING=0 CELLSPACING=0 WIDTH=50%>\n";
      print "<CAPTION><H3>QUERY RESULT TABLE</H3></CAPTION>\n";
      print "<TR>\n";
      for($i = 0; $i < $num_attr; $i++)
        {
         print "<TH>\n";
         print "$attributeAtoms[$i]" ;
         print "</TH>\n";
        }

      print "</TR>\n";
      $oradrh = DBI->install_driver( 'Oracle' );

      $dbh = DBI->connect( 'test', 'xtan', 'xinhong', 'Oracle' );

      $sth = $dbh->prepare( $sql );
```

```
$sth->execute;

 while ( @row = $sth->fetchrow )
   {
    if ($arg{"attrib2"})
       {
        if ( length($row[$non_attr_num]) > 4 )
          {
           $row[$non_attr_num] = substr($row[$non_attr_num],0,4);
          }
       }

   if ($arg{"attrib1"} && $stat1 eq "avg")
     {
      $ind = $non_attr_num + $attr_count - 1;
      if ( length($row[$ind]) > 4 )
         {
          $row[$ind] = substr($row[$ind],0,4);
         }
     }

   print "<TR>\n";
   for($j = 0; $j < $num_attr; $j++)
      {
       print "<TD ALIGN=center>\n";
       print "$row[$j]" ;
       print "</TD>\n";
      }
   print "</TR>\n";
  }


print "</TABLE>\n";
print "</CENTER>\n";

$sth->finish;

die unless $dbh;

$dbh->disconnect;

exit;
}
elsif ( $sg eq "station")
{

 print "<CENTER>\n";
 print "<TABLE BORDER CELLPADDING=0 CELLSPACING=0 WIDTH=50%>\n";
 print "<CAPTION><H3>QUERY RESULT TABLE</H3></CAPTION>\n";
 print "<TR>\n";
 for($i = 0; $i < $num_attr_s; $i++)
   {
    print "<TH>\n";
    print "$attributeAtoms_s[$i]" ;
    print "</TH>\n";
   }

 print "</TR>\n";
 $oradrh = DBI->install_driver( 'Oracle' );
```

```
  $dbh = DBI->connect( 'test', 'xtan', 'xinhong', 'Oracle' );

  $sth = $dbh->prepare( $sql_s );

  $sth->execute;

   while ( @row = $sth->fetchrow )
     {
      if ($arg{"attrib2"})
         {
          if ( length($row[$non_attr_num_s]) > 4 )
            {
             $row[$non_attr_num_s] = substr($row[$non_attr_num_s],0,4);
            }
         }

     if ($arg{"attrib1"} && $stat1 ne "none")
       {
        $ind = $non_attr_num_s + $attr_count - 1;
        if ( length($row[$ind]) > 4 )
           {
            $row[$ind] = substr($row[$ind],0,4);
           }
       }

     print "<TR>\n";
     for($j = 0; $j < $num_attr_s; $j++)
        {
         print "<TD ALIGN=center>\n";
         print "$row[$j]" ;
         print "</TD>\n";
        }
     print "</TR>\n";
    }

  print "</TABLE>\n";
  print "</CENTER>\n";

  $sth->finish;

  die unless $dbh;

  $dbh->disconnect;

  exit;
 }
else
  {

   $oradrh = DBI->install_driver( 'Oracle' );
   $dbh = DBI->connect( 'test', 'xtan', 'xinhong', 'Oracle' );

   #detector table
   print "<CENTER>\n";
   print "<TABLE BORDER CELLPADDING=0 CELLSPACING=0 WIDTH=50%>\n";
   print "<CAPTION><H3>QUERY RESULT TABLE FOR DETECTOR</H3></CAPTION>\n";
   print "<TR>\n";
```

```
for($i = 0; $i < $num_attr; $i++)
  {
   print "<TH>\n";
   print "$attributeAtoms[$i]" ;
   print "</TH>\n";
  }

print "</TR>\n";


$sth = $dbh->prepare( $sql );
$sth->execute;

 while ( @row = $sth->fetchrow )
   {
    if ($arg{"attrib2"})
       {
        if ( length($row[$non_attr_num]) > 4 )
          {
           $row[$non_attr_num] = substr($row[$non_attr_num],0,4);
          }
       }

   if ($arg{"attrib1"} && $stat1 eq "avg")
     {
      $ind = $non_attr_num + $attr_count - 1;
      if ( length($row[$ind]) > 4 )
         {
          $row[$ind] = substr($row[$ind],0,4);
         }
     }

   print "<TR>\n";
   for($j = 0; $j < $num_attr; $j++)
      {
       print "<TD ALIGN=center>\n";
       print "$row[$j]" ;
       print "</TD>\n";
      }
   print "</TR>\n";
  }


print "</TABLE>\n";
print "</CENTER>\n";

$sth->finish;

print "<CENTER>\n";
 print "<TABLE BORDER CELLPADDING=0 CELLSPACING=0 WIDTH=50%>\n";
 print "<CAPTION><H3>QUERY RESULT TABLE FOR STATION</H3></CAPTION>\n";
 print "<TR>\n";
 for($i = 0; $i < $num_attr_s; $i++)
   {
    print "<TH>\n";
    print "$attributeAtoms_s[$i]" ;
    print "</TH>\n";
   }
```

```
    print "</TR>\n";

   $sth = $dbh->prepare( $sql_s );

    $sth->execute;

     while ( @row = $sth->fetchrow )
       {
        if ($arg{"attrib2"})
           {
            if ( length($row[$non_attr_num_s]) > 4 )
              {
               $row[$non_attr_num_s] = substr($row[$non_attr_num_s],0,4);
              }
           }

       if ($arg{"attrib1"} && $stat1 ne "none")
         {
          $ind = $non_attr_num_s + $attr_count - 1;
          if ( length($row[$ind]) > 4 )
             {
              $row[$ind] = substr($row[$ind],0,4);
             }
         }

       print "<TR>\n";
       for($j = 0; $j < $num_attr_s; $j++)
          {
           print "<TD ALIGN=center>\n";
           print "$row[$j]" ;
           print "</TD>\n";
          }
       print "</TR>\n";
      }


   print "</TABLE>\n";
   print "</CENTER>\n";

   $sth->finish;


   die unless $dbh;

   $dbh->disconnect;

   exit;
  }
 }

elsif ( $arg{"action"} eq "Submit as Metrix" )
 {

  if ( $sg eq "detector" )
    {
     $rows_in_det = 0;
     %timeid = ();
     %detid = ();
     @valuelist = ();
```

```
    print "<CENTER>\n";
    print "<TABLE BORDER CELLPADDING=0 CELLSPACING=0 WIDTH=50%>\n";
    print "<CAPTION><H3>QUERY RESULT TABLE</H3></CAPTION>\n";
    print "<TR>\n";

    $oradrh = DBI->install_driver( 'Oracle' );

    $dbh = DBI->connect( 'test', 'xtan', 'xinhong', 'Oracle' );

    $sth = $dbh->prepare( $sql );

    $sth->execute;
   # print $sql;

     while ( @row = $sth->fetchrow )
       {
        if ($arg{"attrib2"})
           {
            if ( length($row[$non_attr_num]) > 4 )
              {
               $row[$non_attr_num] = substr($row[$non_attr_num],0,4);
              }
           }

       if ($arg{"attrib1"} && $stat1 eq "avg")
         {
          $ind = $non_attr_num + $attr_count - 1;
          if ( length($row[$ind]) > 4 )
             {
              $row[$ind] = substr($row[$ind],0,4);
             }
         }

       $timekey = join(" ", @row[0..($time_num - 1)]);
       $timeid{$timekey} = 0;
       $detid{$row[$time_num]} = 0;
       push(@valuelist, join(" ",@row[($time_num + 1)..($num_attr - 1)]));
       $rows_in_det++;
      }

   foreach $com_key (sort keys(%timeid))
          { push(@timeid, $com_key); }
   foreach $com_key (sort keys(%detid))
          { push(@detid, $com_key); }

   $det_num = $rows_in_det / $num_interval;


#first row
   for ( $i = 0; $i < $time_num; $i++)
    {
     print "<TH>\n";
     print "$attributeAtoms[$i]";
     print "</TH>\n";
    }
   for ( $i = 0; $i < $det_num; $i++)
    {
```

```
        for ( $j = 0; $j < $attr_count; $j++)
            {
             print "<TH>\n";
             print "det".$detid[$i].$st.$attributeAtoms[$time_num + $j + 1];
             print "</TH>\n";
            }
       }

      print "</TR>";
      for ( $i = 0; $i < $num_interval; $i++)
       {
         print "<TR>";
         @datetime = split(/ /, $timeid[$i]);
         for ( $j = 0; $j < $time_num; $j++)
            {
             print "<TD ALIGN=CENTER>\n";
             print $datetime[$j];
             print "</TD>\n";
            }
         $start = $i * $det_num;
         $end = $start + $det_num - 1;

         @onerow = @valuelist[$start..$end];

         for ( $j = 0; $j < $det_num; $j++)
            {
             @attribute = split(/ /, $onerow[$j]);
             for ($k = 0; $k < $attr_count; $k++)
                 {
                   print "<TD ALIGN=CENTER>\n";
                   print $attribute[$k];
                   print "</TD>\n";
                 }
            }
         print "</TR>";
       }


      print "</TABLE>\n";
      print "</CENTER>\n";

      $sth->finish;

      die unless $dbh;

      $dbh->disconnect;

      exit;

     }


elsif ( $sg eq "station")
     {

      $rows_in_stat = 0;
      %timeid = ();
      %statid = ();
```

```
@valuelist = ();

print "<CENTER>\n";
print "<TABLE BORDER CELLPADDING=0 CELLSPACING=0 WIDTH=50%>\n";
print "<CAPTION><H3>QUERY RESULT TABLE</H3></CAPTION>\n";
print "<TR>\n";

$oradrh = DBI->install_driver( 'Oracle' );

$dbh = DBI->connect( 'test', 'xtan', 'xinhong', 'Oracle' );

$sth = $dbh->prepare( $sql_s );

$sth->execute;
while ( @row = $sth->fetchrow )
   {
    if ($arg{"attrib2"})
       {
        if ( length($row[$non_attr_num_s]) > 4 )
          {
           $row[$non_attr_num_s] = substr($row[$non_attr_num_s],0,4);
          }
       }

   if ($arg{"attrib1"} && $stat1 eq "avg")
     {
      $ind = $non_attr_num_s + $attr_count - 1;
      if ( length($row[$ind]) > 4 )
         {
          $row[$ind] = substr($row[$ind],0,4);
         }
     }

   $timekey = join(" ", @row[0..($time_num - 1)]);
   $timeid{$timekey} = 0;
   $statid{$row[$time_num]} = 0;
   push(@valuelist, join(" ",@row[($time_num + 1)..($num_attr_s - 1)]));
   $rows_in_stat++;
  }
 foreach $com_key (sort keys(%timeid))
       { push(@timeid, $com_key); }
foreach $com_key (sort keys(%statid))
       { push(@statid, $com_key); }

$stat_num = $rows_in_stat / $num_interval;


#first row
     for ( $i = 0; $i < $time_num; $i++)
      {
       print "<TH>\n";
       print "$attributeAtoms_s[$i]";
       print "</TH>\n";
      }
     for ( $i = 0; $i < $stat_num; $i++)
      {
       for ( $j = 0; $j < $attr_count; $j++)
           {
```

```
            print "<TH>\n";
            print "stat".$statid[$i].$st.$attributeAtoms_s[$time_num + $j + 1];
            print "</TH>\n";
           }
     }


    for ( $i = 0; $i < $num_interval; $i++)
     {
       print "<TR>";
       @datetime = split(/ /, $timeid[$i]);
       for ( $j = 0; $j < $time_num; $j++)
          {
           print "<TD ALIGN=CENTER>\n";
           print $datetime[$j];
           print "</TD>\n";
          }
       $start = $i * $stat_num;
       $end = $start + $stat_num - 1;

       @onerow = @valuelist[$start..$end];

       for ( $j = 0; $j < $stat_num; $j++)
          {
           @attribute = split(/ /, $onerow[$j]);
           for ($k = 0; $k < $attr_count; $k++)
               {
                 print "<TD ALIGN=CENTER>\n";
                 print $attribute[$k];
                 print "</TD>\n";
               }
          }
       print "</TR>";
     }


     print "</TABLE>\n";
    print "</CENTER>\n";

    $sth->finish;

    die unless $dbh;

    $dbh->disconnect;

    exit;

    }

else
    {
     %detec_stat = ();
     %timelist = ();
     $rows_in_det = 0;
     $rows_in_stat = 0;


     $coma = ",";
```

```
$oradrh = DBI->install_driver( 'Oracle' );

$dbh = DBI->connect( 'test', 'xtan', 'xinhong', 'Oracle' );

$sth = $dbh->prepare( $sql );

$sth->execute;


while ( @row = $sth->fetchrow )
   {
    if ($arg{"attrib2"})
       {
        if ( length($row[$non_attr_num]) > 4 )
          {
           $row[$non_attr_num] = substr($row[$non_attr_num],0,4);
          }
       }

   if ($arg{"attrib1"} && $stat1 eq "avg")
     {
      $ind = $non_attr_num + $attr_count - 1;
      if ( length($row[$ind]) > 4 )
         {
          $row[$ind] = substr($row[$ind],0,4);
         }
     }
    $com_key = join(" ", @row[0..($non_attr_num - 1)]);
    $detec_stat{$com_key} = join(" ", @row[$non_attr_num..($num_attr - 1)]);
    $rows_in_det+= 1;
   }

$sth->finish;

# print "rows_in_det", $rows_in_det;

$sth_s = $dbh->prepare( $sql_s );

$sth_s->execute;


while ( @row = $sth_s->fetchrow )
   {
    $timestamp = join(" ", @row[0..($time_num - 1)]);
    $timelist{$timestamp} = $coma;
    $com_key = join(" "; @row[0..$time_num]);
    if ($arg{"attrib2"})
       {
        if ( length($row[$non_attr_num_s]) > 4 )
          {
           $row[$non_attr_num_s] = substr($row[$non_attr_num_s],0,4);
          }
       }

   if ($arg{"attrib1"} && $stat1 ne "none")
     {
      $ind = $non_attr_num_s + $attr_count - 1;
      if ( length($row[$ind]) > 4 )
```

```
          {
           $row[$ind] = substr($row[$ind],0,4);
          }
      }
     $detec_stat{$com_key} = join(" ", @row[$non_attr_num_s..($num_attr_s - 1)]);
     $rows_in_stat+= 1;
    }


 $sth_s->finish;

 die unless $dbh;

 $dbh->disconnect;
#    print "rows_in_stat", $rows_in_stat;

 $num_det = $rows_in_det / $num_interval;
 $num_stat = $rows_in_stat / $num_interval;

# print "num_det", $num_det,"num_stat", $num_stat;

 @timekey = sort keys(%timelist);

 @valuelist = ();
 foreach $com_key (sort keys(%detec_stat))
     {
      push(@keylist, $com_key);
      push(@valuelist, $detec_stat{$com_key});
     }

 print "<CENTER>\n";
 print "<TABLE BORDER CELLPADDING=0 CELLSPACING=0 WIDTH=50%>\n";
 print "<CAPTION><H3>QUERY RESULT TABLE</H3></CAPTION>\n";
 print "<TR>\n";
 for($i = 0; $i < $num_attr; $i++)
   {
    if($attributeAtoms[$i] =~ /\./)
     {
      @subattr = split(/\./, $attributeAtoms[$i]);
      $num = @subattr;
      $attributeAtoms[$i] = $subattr[ $num - 1];
     }
   }

  for($i = 0; $i < $time_num; $i++)
   {
    print "<TH>\n";
    print "$attributeAtoms[$i]" ;
    print "</TH>\n";
   }

 @idlist = ();


 for($i = 0; $i < ($num_det + $num_stat); $i++)
     {
      @fields = split(/ /, $keylist[$i]);
      if($#fields == ($non_attr_num_s - 1))
```

```
          {
           $statid = pop(@fields);

           for($j = $attr_count ; $j > 0; $j--)
               {
                $temp = "stat".$statid.$attributeAtoms[($num_attr_s  - $j + 1)];
                print "<TH ALIGN=center>\n";
                print "stat".$statid.$st.$attributeAtoms[($num_attr_s  - $j + 1)];
                print "</TH>\n";
               }
          }
        else
          {
           $detid = pop(@fields);

           for($j = $attr_count ; $j > 0; $j--)
               {
                print "<TH ALIGN=center>\n";
                print "det".$detid.$st.$attributeAtoms[($num_attr_s  - $j + 1)];
                print "</TH>\n";
               }

          }
      }

  print "</TR>\n";
  for($i = 0; $i < $num_interval; $i++)
      {
       print "<TR>\n";
       @datetime = split(/ /,$timekey[$i]);
       for($j = 0; $j < $time_num; $j++)
          {
           print "<TD ALIGN=center>";
           print $datetime[$j];
           print "</TD>\n";
          }

        $start = $i * ($num_det + $num_stat);
        $end = $start + $num_det + $num_stat - 1;
        #print " start =$start end = $end ";
        @onerow = @valuelist[$start..$end];
       for($j = 0; $j < ($num_det + $num_stat); $j++)
          {
           @attribute = split(/ /, $onerow[$j]);
           for($k = 0; $k < $attr_count; $k++)
               {
                print "<TD ALIGN=center>";
                print $attribute[$k];
                print "</TD>\n";
               }
         }
        print "</TR>\n";
   }
   print "</TABLE>";
   print "</CENTER>";


#
```

```
      }
  }
elsif ( $arg{"action"} eq "Show SQL" )
   {
    if ( $sg ne "station")
       {
        print "<CENTER>\n";
        print "<TABLE BORDER >\n";
        print "<TR>\n";
        print "<TD>\n";
        print "<CENTER> <H1>SQL STATEMENT</H1></CENTER>\n";
        print "</TD>\n";
        print "</TR>\n";
        print "<TR>\n";
        print "<TD>\n";
        print "$sql \n";
        print "</TD>\n";
        print "</TR>\n";
        print "</TABLE>\n";
        print "</CENTER>\n";
       }
    if ( $sg ne "detector"  )
       {
        print "<CENTER>\n";
        print "<TABLE BORDER >\n";
        print "<TR>\n";
        print "<TD>\n";
        print "<CENTER> <H1>SQL STATEMENT</H1></CENTER>\n";
        print "</TD>\n";
        print "</TR>\n";
        print "<TR>\n";
        print "<TD>\n";
        print "$sql_s \n";
        print "</TD>\n";
        print "</TR>\n";
        print "</TABLE>\n";
        print "</CENTER>\n";
       }
    }
}

sub read_input
{
    local ($buffer, @pairs, $pair, $name, $value, %FORM);
    # Read in text
    $ENV{'REQUEST_METHOD'} =~ tr/a-z/A-Z/;
    if ($ENV{'REQUEST_METHOD'} eq "POST")
    {
        read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});
    } else
    {
        $buffer = $ENV{'QUERY_STRING'};
    }
    # Split information into name/value pairs
    @pairs = split(/&/, $buffer);
    foreach $pair (@pairs)
    {
        ($name, $value) = split(/=/, $pair);
        $value =~ tr/+/ /;
```

```
        $value =~ s/%(..)/pack("C", hex($1))/eg;
        $FORM{$name} = $value;
    }
    %FORM;
}
```

APPENDIX G

Benchmark Queries

The appendix lists the fifteen benchmark queries with their corresponding SQL statement, as well as the GUI formulation.

Q1. Get 5-min Volume, occupancy for detector ID = 10 on Oct. 1st, 1997 from 7am to 8am

```
SELECT readdate, time, xtan.fivemin.detector, occupancy, volume
FROM xtan.fivemin, xtan.datetime
WHERE ReadDate = to_date('01-OCT-97', 'DD-MON-YYYY')
AND time BETWEEN '0705' AND '0800'
AND xtan.fivemin.Detector = '10'
AND xtan.fivemin.timeid = datetime.timeid
```

QUERY RESULT TABLE

readdate	time	detector	occupancy	volume
1-Oct-97	0705	10	1	21
1-Oct-97	0710	10	1	24
1-Oct-97	0715	10	2	29
1-Oct-97	0720	10	1	15
1-Oct-97	0725	10	0	13
1-Oct-97	0730	10	1	23
1-Oct-97	0735	10	0	11
1-Oct-97	0740	10	1	19
1-Oct-97	0745	10	0	14
1-Oct-97	0750	10	1	19
1-Oct-97	0755	10	1	17
1-Oct-97	0800	10	0	16

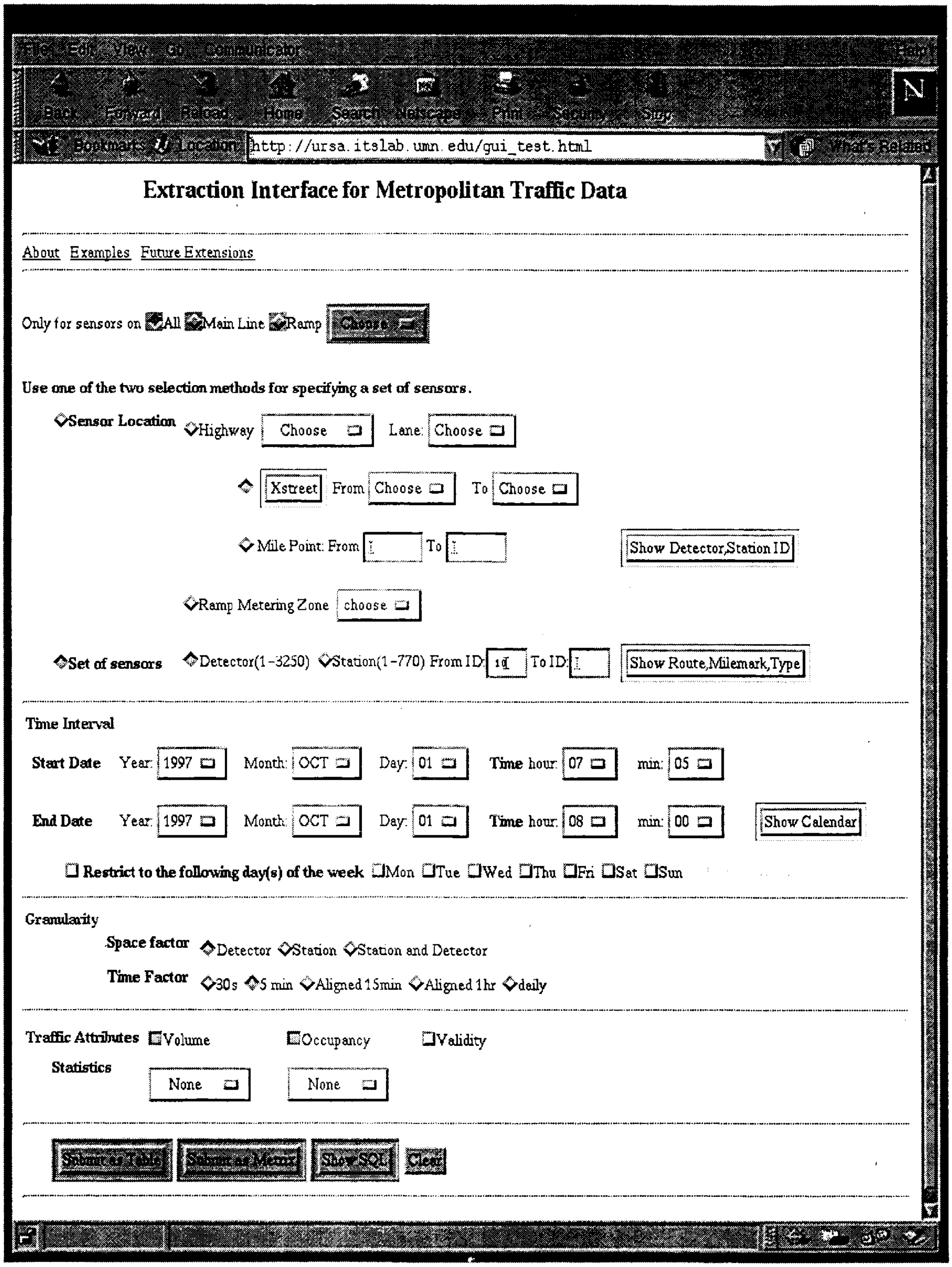

Figure 3: GUI specification for Q1

Q2. Get 5-min Volume, occupancy for detector ID = 8 on Oct. 1st, 1997

```
SELECT readdate, time, xtan.fivemin.detector, occupancy, volume
FROM xtan.fivemin, xtan.datetime
WHERE ReadDate = to_date('01-OCT-97','DD-MON-YYYY')
AND time BETWEEN '0005' AND '2400'
AND xtan.fivemin.Detector = '8'
AND xtan.fivemin.timeid = datetime.timeid
```

QUERY RESULT TABLE

readdate	time	detector	occupancy	volume
1-Oct-97	0005	8	1	14
1-Oct-97	0010	8	2	20
1-Oct-97	0015	8	1	12
1-Oct-97	0020	8	0	10
1-Oct-97	0025	8	1	11
1-Oct-97	0030	8	1	12
1-Oct-97	0035	8	1	10
1-Oct-97	0040	8	1	14
1-Oct-97	0045	8	0	7
1-Oct-97	0050	8	1	14
1-Oct-97	0055	8	1	9
1-Oct-97	0100	8	0	10
1-Oct-97	0105	8	0	8
1-Oct-97	0110	8	1	11
1-Oct-97	0115	8	1	10
1-Oct-97	0120	8	0	5
1-Oct-97	0125	8	0	9
1-Oct-97	0130	8	1	11
1-Oct-97	0135	8	0	6
1-Oct-97	0140	8	0	10
1-Oct-97	0145	8	0	6
1-Oct-97	0150	8	0	4
1-Oct-97	0155	8	0	7
1-Oct-97	0200	8	0	5
1-Oct-97	0205	8	0	12
1-Oct-97	0210	8	0	6
1-Oct-97	0215	8	0	8

Q3. Get 5-min Volume, occupancy for detector ID from 1 to 5 on Oct. 1st, 1997 from 7am to 8a

```
SELECT readdate, time, xtan.fivemin.detector, occupancy, volume
FROM xtan.fivemin, xtan.datetime
WHERE ReadDate = to_date('01-OCT-97','DD-MON-YYYY')
AND time BETWEEN '0705' AND '0800'
AND xtan.fivemin.Detector BETWEEN '1' AND '5'
AND xtan.fivemin.timeid = datetime.timeid
```

QUERY RESULT TABLE

readdate	time	detector	occupancy	volume
1-Oct-97	0705	1	0	0
1-Oct-97	0705	2	0	8
1-Oct-97	0705	3	1	14
1-Oct-97	0705	4	0	4
1-Oct-97	0705	5	6	27
1-Oct-97	0710	1	0	0
1-Oct-97	0710	2	1	12
1-Oct-97	0710	3	1	15
1-Oct-97	0710	4	1	7
1-Oct-97	0710	5	9	29
1-Oct-97	0715	1	0	0
1-Oct-97	0715	2	1	10
1-OcT-97	0715	3	3	22
1-Oct-97	0715	4	1	9
1-oct-97	0715	5	12	40
1-Oct-97	0720	1	0	0
1-Oct-97	0720	2	2	17
1-Oct-97	0720	3	1	14
1-Oct-97	0720	4	1	7
1-Oct-97	0720	5	11	30
1-Oct-97	0725	1	0	0
1-Oct-97	0725	2	1	12
1-Oct-97	0725	3	2	15
1-Oct-97	0725	4	1	7
1-Oct-97	0725	5	13	39
1-Oct-97	0730	1	0	0
1-Oct-97	0730	2	1	10
1-Oct-97	0730	3	1	13
1-Oct-97	0730	4	0	4

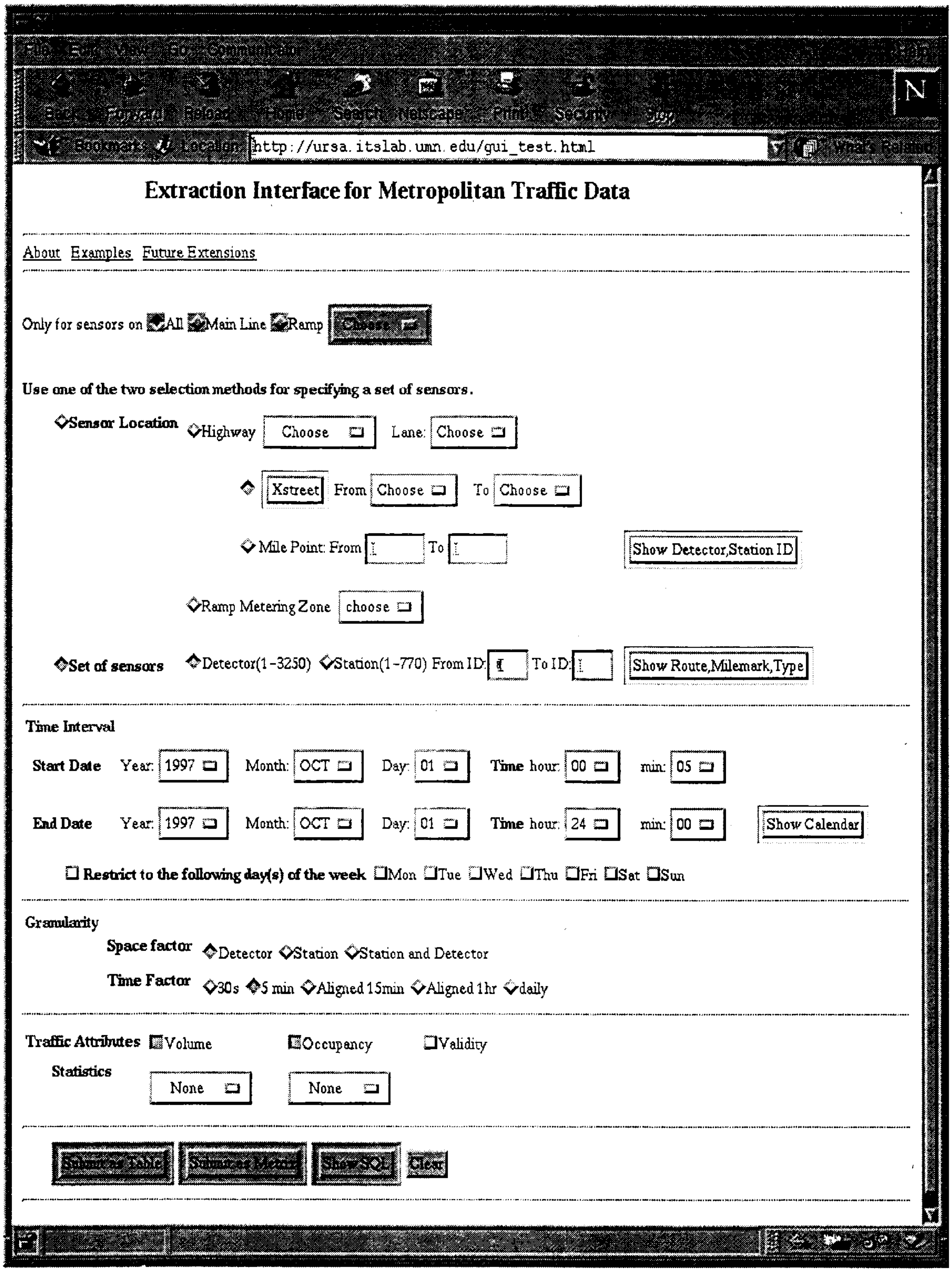

Figure 4: GUI specification for Q2

Q4. Get 5-min Volume, occupancy for station ID = 9 on Oct. 1st, 1997 from 7am to 8am

```
SELECT readdate, time, xtan.v_stat_five.station, occupancy, volume
FROM xtan.v_stat_five
WHERE ReadDate = to_date('01-OCT-97','DD-MON-YYYY')
AND time BETWEEN '0705' AND '0800'
AND xtan.v_stat_five.station = '9'
```

QUERY RESULT TABLE

```
readdate time station occupancy volume
1-Oct-97         0705 9 10.75 537
1-Oct-97 7010 9 10.5 545
1-Oct-97 0715 9 9.25 489
1-Oct-97 0720 9 10.5 525
1-Oct-97 0725 9 9.75 503
1-Oct-97 0730 9 10.5 471
1-Oct-97 0735 9 23.5 501
1-Oct-97 0740 9 13 462
1-Oct-97 0745 9 9.75 412
1-Oct-97 0750 9 8.75 422
1-Oct-97 0755 9 9.25 455
1-Oct-97 0800 9 10.5 513
```

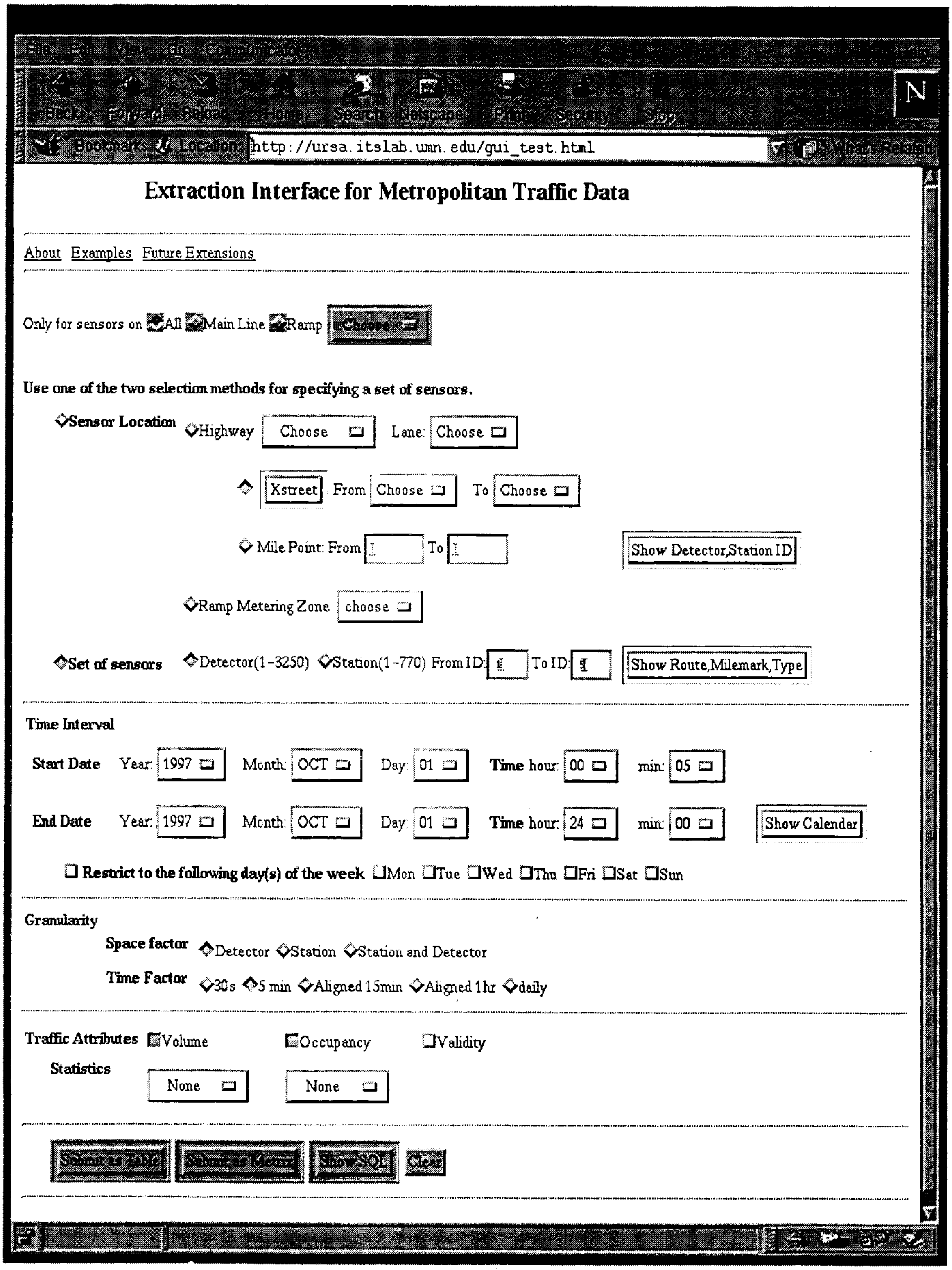

Figure 5: GUI specification for Q3

Q5. Get 5-min maximum Volume, maximum occupancy for station ID = 40 on Oct. 1st, 1997 from 6:

```
SELECT time, xtan.v_stat_five.station, max(occupancy), max(volume)
FROM xtan.v_stat_five
WHERE ReadDate = to_date('01-OCT-97','DD-MON-YYYY')
AND time BETWEEN '0605' AND '0700'
AND xtan.v_stat_five.station = '40'
GROUP BY xtan.v_stat_five.station, time
```

QUERY RESULT TABLE

```
time station        max(occupancy) max(volume)
0605 40 10 347
0610 40 11 386
0615 40 10.666 381
0620 40 11.666 404
0625 40 11.666 399
0630 40 13.666 408
0635 40 12.666 408
0640 40 12.666 421
0645 40 11.666 383
0650 40 12.333 383
0655 40 13 429
0700 40 12.666 406
```

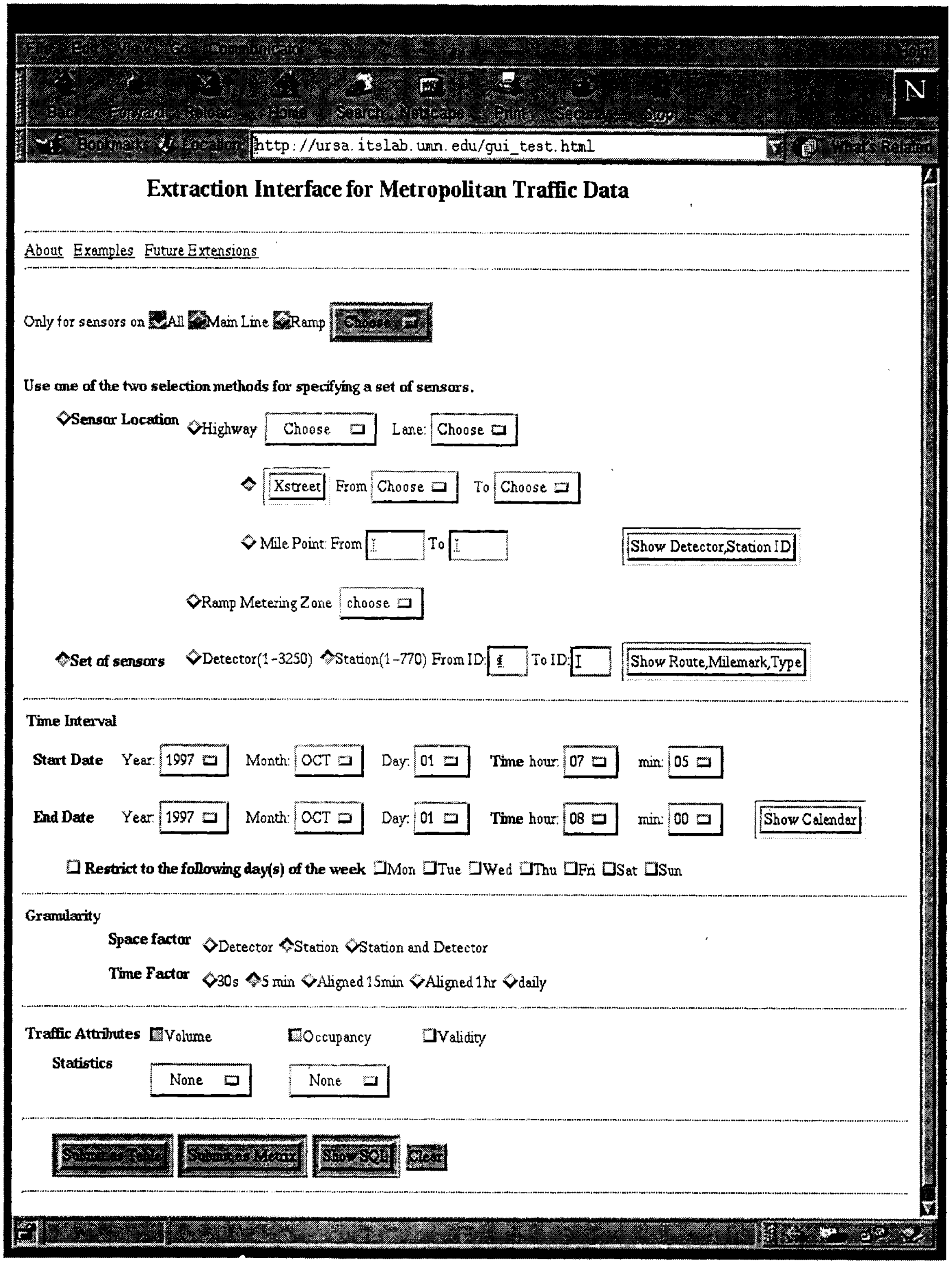

Figure 6: GUI specification for Q4

Q6. Get the sum of hourly volume for station ID = 4 from Oct. 1st, 1997 to Oct. 5th , 1997

```
SELECT hour, xtan.v_stat_hour.station, sum(volume)
FROM xtan.v_stat_hour
WHERE ReadDate BETWEEN to_date('01-OCT-97','DD-MON-YYYY')  AND to_date('05-OCT-97','DD-MON-
AND hour BETWEEN '00' AND '24'
AND xtan.v_stat_hour.station = '4'
GROUP BY xtan.v_stat_hour.station, hour
```

QUERY RESULT TABLE

hour	station	sum(volume)
1	4	4342
2	4	2341
3	4	1774
4	4	2484
5	4	7031
6	4	16519
7	4	19496
8	4	18584
9	4	18693
10	4	20662
11	4	26735
12	4	28870
13	4	29678
14	4	29665
15	4	32154
16	4	31832
17	4	30348
18	4	28636
19	4	22632
20	4	19269
21	4	17823
22	4	14159
23	4	9932
24	4	533

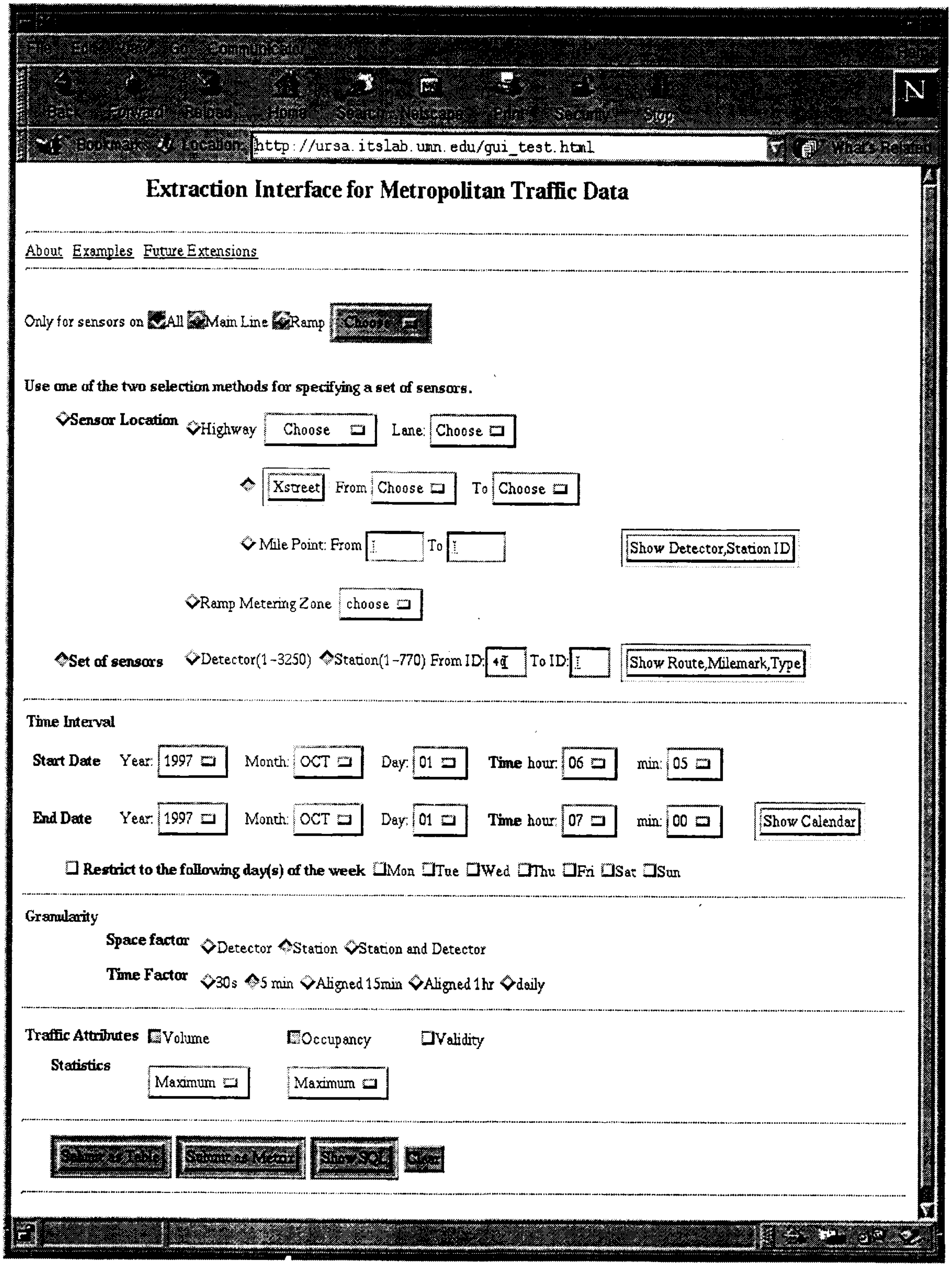

Figure 7: GUI specification for Q5

Q7. Get 5-min average volume, maximum occupancy for station ID = 20 on Mondays in Oct, 199

```
SELECT time, xtan.v_stat_five.station, max(occupancy), avg(volume)
FROM xtan.v_stat_five
WHERE to_char(readdate, 'MON-RR') = 'OCT-97'
AND time BETWEEN '0805' AND '0815'
AND DayofWeek IN ('2')
AND xtan.v_stat_five.station = '20'
GROUP BY xtan.v_stat_five.station, time
```

QUERY RESULT TABLE

```
times tation max(occupancy) avg(volume)
0805 20 20.1 308.33
0810 20 11.65 319.66
0815 20 11.25 306.33
```

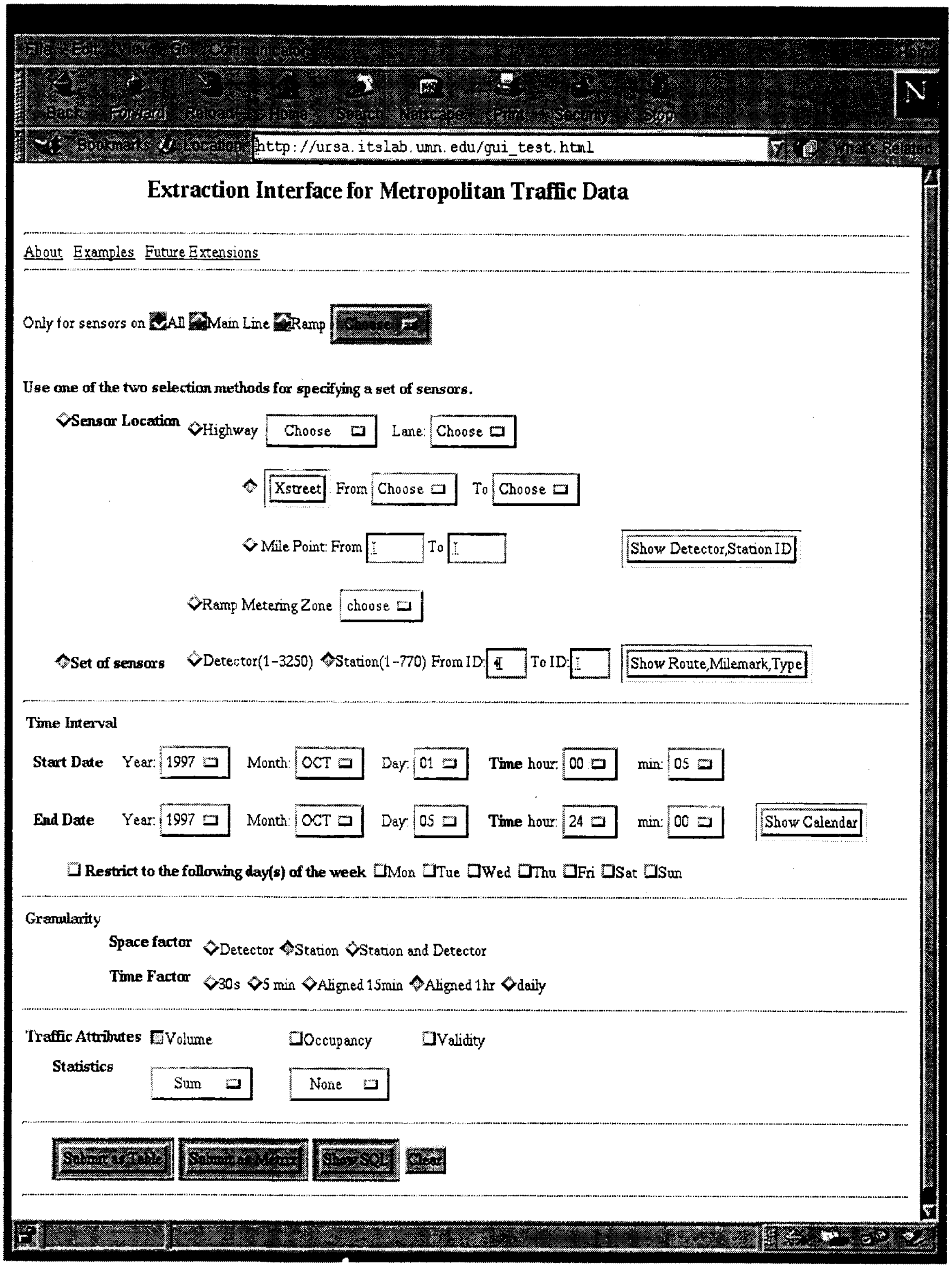

Figure 8: GUI specification for Q6

Q8. Get hourly volume for station ID = 40 on Monday and Tuesdays in Oct, 1997

```
SELECT readdate, hour, xtan.v_stat_hour.station, volume
FROM xtan.v_stat_hour
WHERE to_char(readdate, 'MON-RR') = 'OCT-97'
AND hour BETWEEN '00' AND '24'
AND DayofWeek IN ('2','3')
AND xtan.v_stat_hour.station = '40'
```

QUERY RESULT TABLE

readdate	hour	station	volume
6-Oct-97	1	40	221
6-Oct-97	2	40	30
6-Oct-97	3	40	0
6-Oct-97	4	40	0
6-Oct-97	5	40	1077
6-Oct-97	6	40	4543
6-Oct-97	7	40	4191
6-Oct-97	8	40	4161
6-Oct-97	9	40	3415
6-Oct-97	10	40	2866
6-Oct-97	11	40	2978
6-Oct-97	12	40	3052
6-Oct-97	13	40	3054
6-Oct-97	14	40	3192
6-Oct-97	15	40	3313
6-Oct-97	16	40	3304
6-Oct-97	17	40	3551
6-Oct-97	18	40	2913
6-Oct-97	19	40	1976
6-Oct-97	20	40	1595
6-Oct-97	21	40	1453
6-Oct-97	22	40	959
6-Oct-97	23	40	532
6-Oct-97	24	40	260
7-Oct-97	1	40	180
7-Oct-97	2	40	212
7-Oct-97	3	40	0
7-Oct-97	4	40	0
7-Oct-97	5	40	0
7-Oct-97	6	40	4662
7-Oct-97	7	40	4911
7-Oct-97	8	40	4457

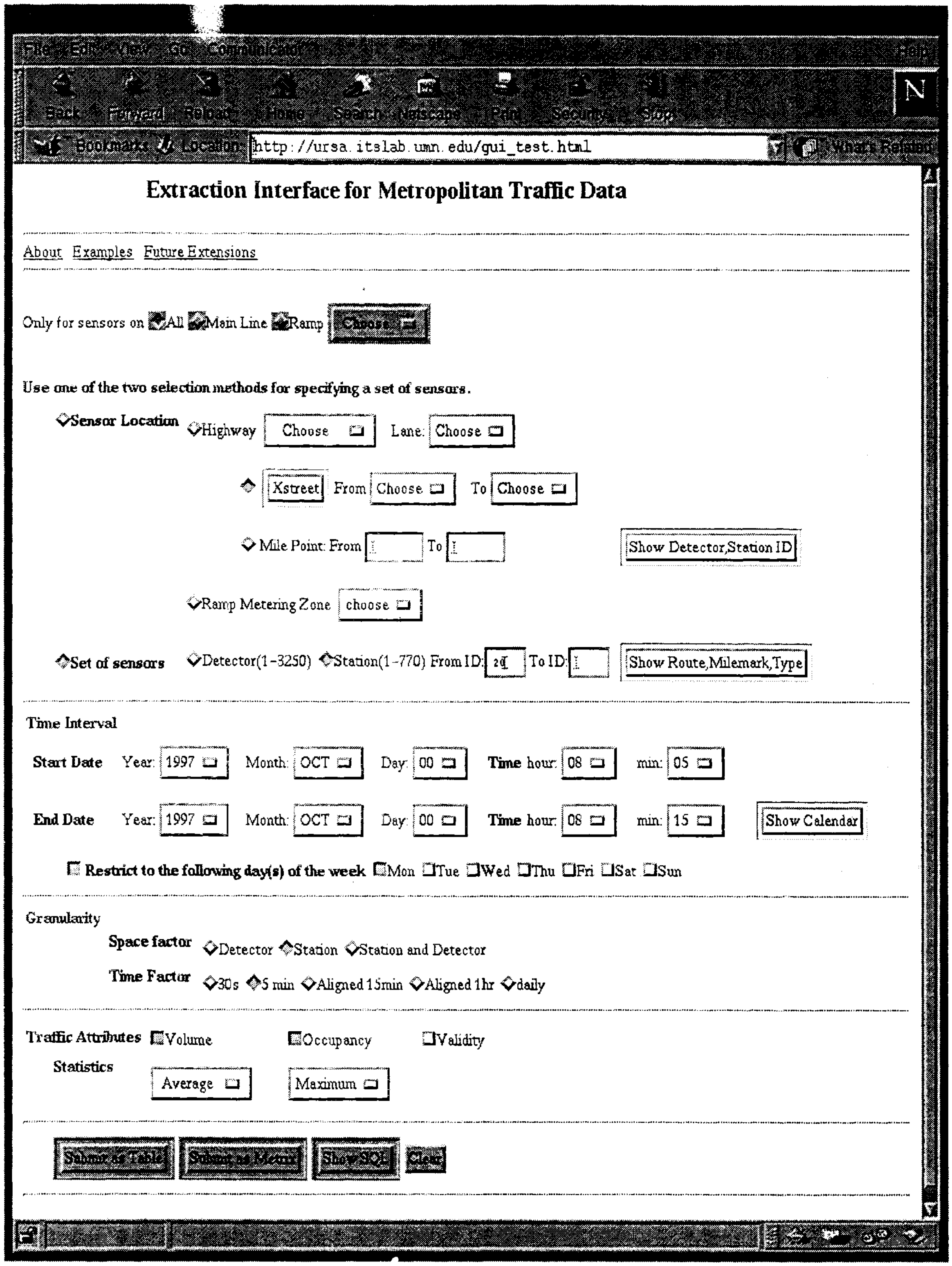

Figure 9: GUI specification for Q7

Q9. Get 5-min volume, occupancy for all detectors in station ID = 24 from 7am to 8am on Oct

```
SELECT readdate, hour, xtan.v_hour.detector, occupancy, volume
FROM xtan.v_hour, xtan.detector
WHERE ReadDate = to_date('01-OCT-97','DD-MON-YYYY')
AND hour BETWEEN '00' AND '24'
AND detector.station = '5'
AND xtan.v_hour.detector = detector.detector
```

QUERY RESULT TABLE

```
readdate time detector occupancy volume
1-Oct-97        0705 242.7 99
1-Oct-97 0705 243 10 136
1-Oct-97 0705 355 0 11
1-Oct-97 0710 242 7 99
1-Oct-97 0710 243 10 129
1-Oct-97 0710 355 1 16
1-Oct-97 0715 242 7 86
1-Oct-97 0715 243 9 127
1-Oct-97 0715 355 1 15
1-Oct-97 0720 242 7 95
1-Oct-97 0720 243 8 119
1-Oct-97 0720 355 1 14
1-Oct-97 0725 242 8 105
1-Oct-97 0725 243 10 141
1-Oct-97 0725 355 1 18
1-Oct-97 0730 242 9 107
1-Oct-97 0730 243 9 123
1-Oct-97 0730 355 0 13
1-Oct-97 0735 242 8 98
1-Oct-97 0735 243 8 122
1-Oct-97 0735 355 0 13
1-Oct-97 0740 242 8 113
1-Oct-97 0740 243 8 121
1-Oct-97 0740 355 0 9
1-Oct-97 0745 242 9 109
1-Oct-97 0745 243 10 136
1-Oct-97 0745 355 1 20
1-Oct-97 0750 242 7 86
1-Oct-97 0750 243 9 123
1-Oct-97 0750 355 0 13
1-Oct-97 0755 242 10 116
1-Oct-97 0755 243 10 142
1-Oct-97 0755 355 1 16
1-Oct-97 0800 242 9 115
1-Oct-97 0800 243 11 130
```

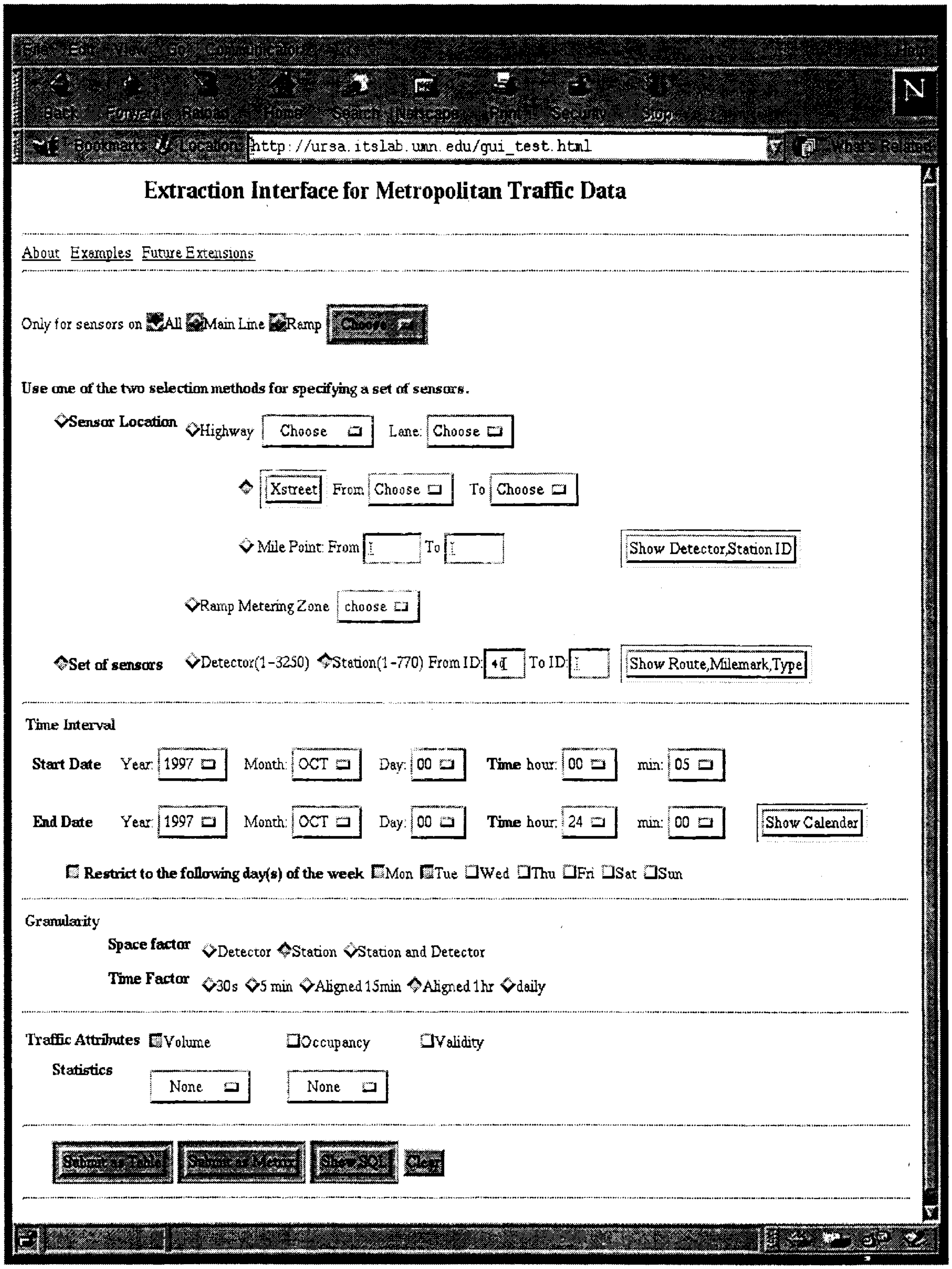

Figure 10: GUI specification for Q8

Q10. Get 5-min volume, occupancy for a set of stations on highway I35W-NB with milepoint be

```
SELECT readdate, time, xtan.detector.station, xtan.p_fivemin.detector, occupancy, volume
FROM xtan.fivemin, xtan.datetime, xtan.detector, xtan.statrdwy
WHERE ReadDate = to_date('01-OCT-97','DD-MON-YYYY')
AND time BETWEEN '0705' AND '0730'
AND statrdwy.route = 'I35W-I'
AND statrdwy.mp >= 0.0
AND statrdwy.mp <= 4.0
AND xtan.fivemin.timeid = datetime.timeid
AND xtan.fivemin.detector = detector.detector
AND detector.station = statrdwy.station
```

```
SELECT readdate, time, xtan.v_stat_five.station, occupancy, volume
FROM xtan.v_stat_five, xtan.statrdwy
WHERE ReadDate = to_date('01-OCT-97','DD-MON-YYYY')
AND time BETWEEN '0705' AND '0730'
AND statrdwy.route = 'I35W-I'
AND statrdwy.mp >= 0.0
AND statrdwy.mp <= 4.0
AND xtan.v_stat_five.station = statrdwy.station
```

QUERY RESULT TABLE

```
Readdate time stat32-occ stat32-vol det258-occ det258-vol det259-occ det259-vol det461-occ
det461-vol
1-Oct-97 0705 7.6666 339 8 117 13 184 2 38
1-Oct-97 0710 14 332 9 112 19 220 18.5 377
1-Oct-97 0715 25 164 3 40 13.666 369 17 127
1-Oct-97 0720 7.6666 312 9 118 13 169 1 25
1-Oct-97 0725 11 262 9 98 13 164 18 339
1-Oct-97 0730 15 155 1 24 16.333 327 22 118
```

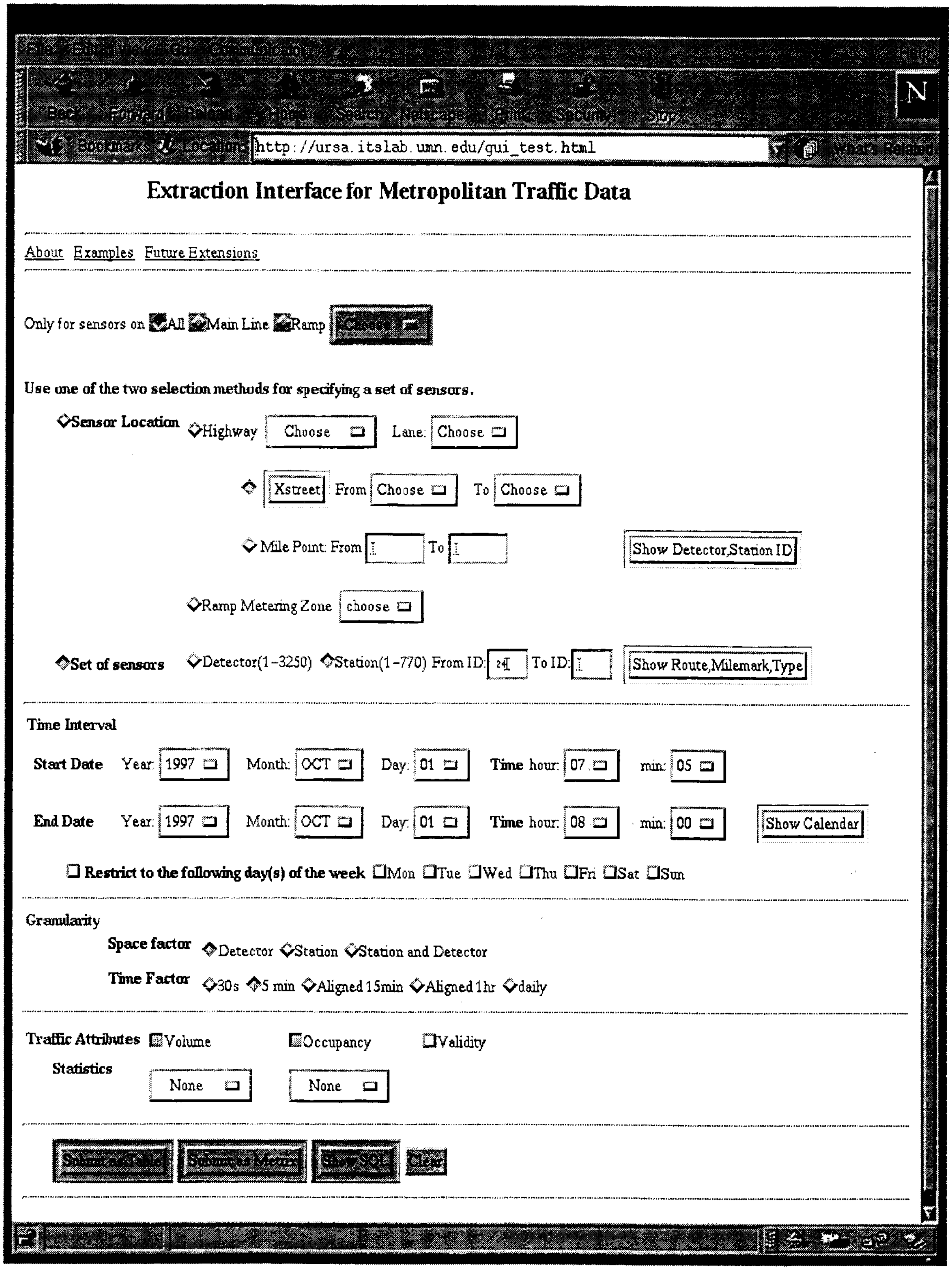

Figure 11: GUI specification for Q9

Q11. Get 5-min volume for a set of stations on highway I35W-NB between Co Rd 42 and Burnsv

```
SELECT readdate, time, xtan.v_stat_five.station, occupancy, volume
FROM xtan.v_stat_five, xtan.statrdwy
WHERE ReadDate = to_date('01-OCT-97','DD-MON-YYYY')
AND time BETWEEN '0005' AND '2400'
AND statrdwy.route = 'I35W-I'
AND statrdwy.mp >= .6
AND statrdwy.mp <= 2.3
AND xtan.v_stat_five.station = statrdwy.station
```

QUERY RESULT TABLE

readdate	time	station	occupancy	volume
1-Oct-97	0005	32	0	19
1-Oct-97	0010	32	0	14
1-Oct-97	0015	32	0.33333	26
1-Oct-97	0020	32	0	14
1-Oct-97	0025	32	0	16
1-Oct-97	0030	32	0	10
1-Oct-97	0035	32	0	12
1-Oct-97	0040	32	0	10
1-Oct-97	0045	32	0	8
1-Oct-97	0050	32	0	12
1-Oct-97	0055	32	0.33333	13
1-Oct-97	0100	32	0	9
1-Oct-97	0105	32	0	14
1-Oct-97	0005	71	0.5	10
1-Oct-97	0010	71	1	13
1-Oct-97	0015	71	1	19
1-Oct-97	0020	71	0.5	11
1-Oct-97	0025	71	0.5	11
1-Oct-97	0030	71	0	5
1-Oct-97	0035	71	0.5	6
1-Oct-97	0040	71	0.5	8
1-Oct-97	0045	71	0	5
1-Oct-97	0050	71	0.5	10
1-Oct-97	0055	71	0.5	9
1-Oct-97	0100	71	0	7
1-Oct-97	0105	71	0.5	8
1-Oct-97	0005	72	0.5	18
1-Oct-97	0010	72	1	22
1-Oct-97	0015	72	1.5	30
1-Oct-97	0020	72	0.5	16
1-Oct-97	0025	72	1	20
1-Oct-97	0030	72	0	16
1-Oct-97	0035	72	0.5	14
1-Oct-97	0040	72	0.5	10

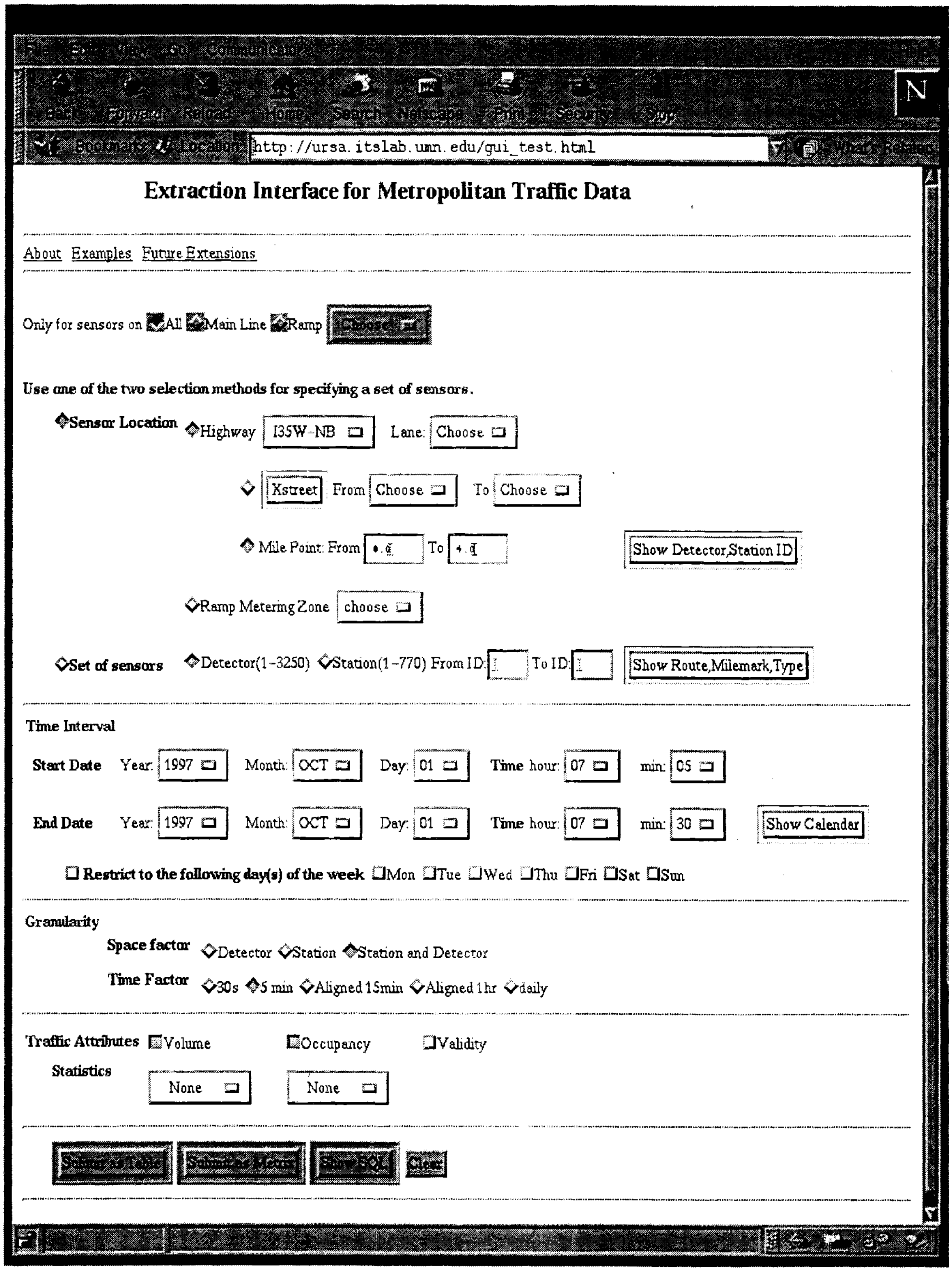

Figure 12: GUI specification for Q10

Q12. Get the average of AM rushhour hourly volume for a set of stations on highway I35W-NB

```
SELECT hour, xtan.v_stat_hour.station, avg(volume)
FROM tan.v_stat_hour, xtan.statrdwy
WHERE ReadDate BETWEEN to_date('01-OCT-97','DD-MON-YYYY')  AND to_date('05-OCT-97','DD-MON-
AND hour BETWEEN '06' AND '09'
AND statrdwy.route = 'I35W-I'
AND statrdwy.mp >= 0.0
AND statrdwy.mp <= 4.0
AND xtan.v_stat_hour.station = statrdwy.station
GROUP BY xtan.v_stat_hour.station, hour
```

QUERY RESULT TABLE

hour	station	avg(volume)
6	32	2233.2
7	32	2270.4
8	32	1861.6
9	32	1507.2
6	33	2446.8
7	33	2553
8	33	2126
9	33	1786
6	35	2916.4
7	35	3069.8
8	35	2740.8
9	35	2218
6	71	1722.4
7	71	1989.8
8	71	1271
9	71	982
6	72	2280
7	72	2508.2
8	72	1921.8
9	72	1598.2

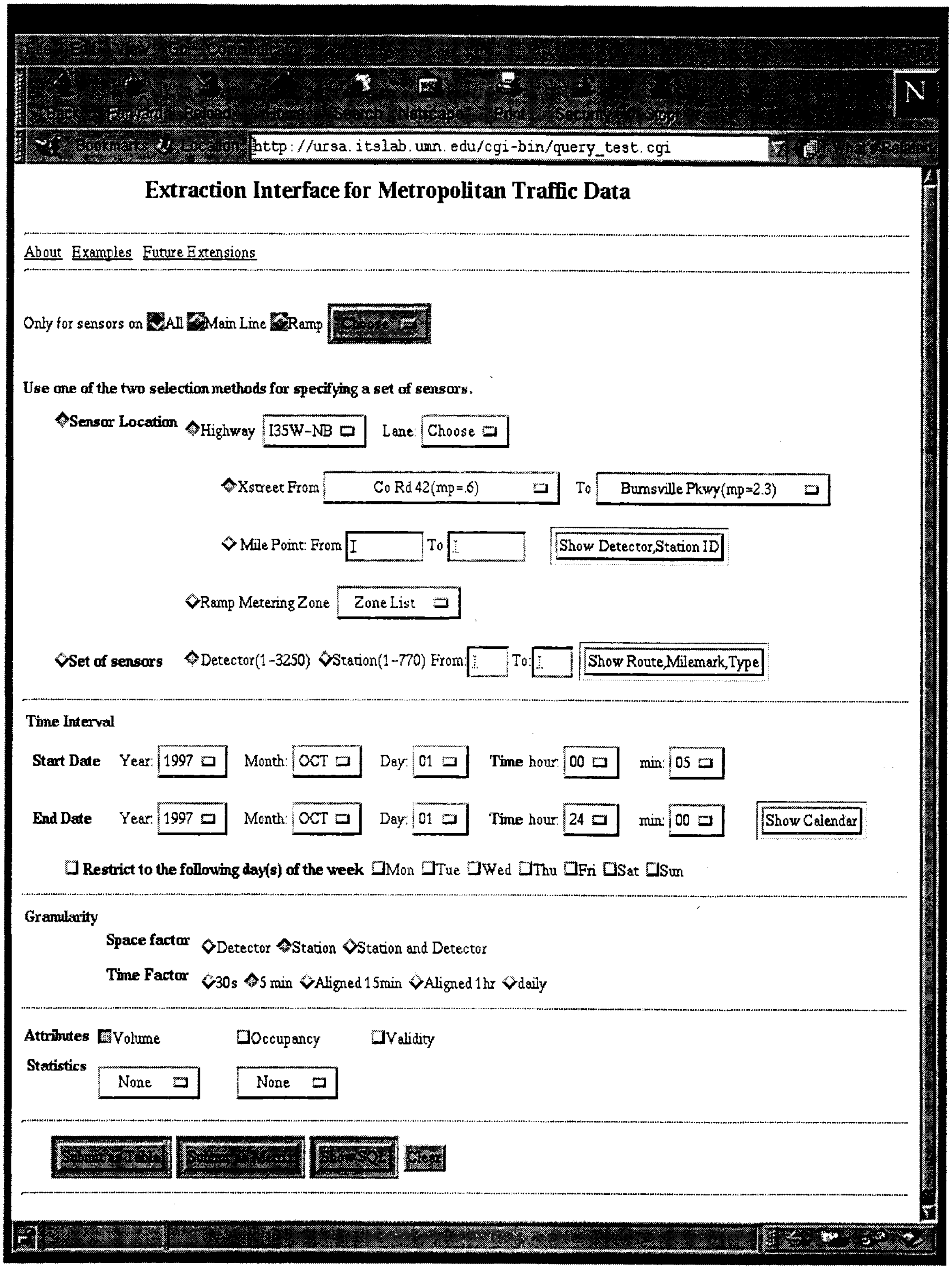

Figure 13: GUI specification for Q11

```
Q13. SELECT readdate, hour, xtan.v_stat_hour.station, volume
FROM xtan.v_stat_hour
WHERE ReadDate = to_date('01-OCT-97','DD-MON-YYYY')
AND hour BETWEEN '06' AND '09'
AND testzone.zonename = '1A'
AND xtan.v_stat_hour.station = testzone.station
```

QUERY RESULT TABLE

readdate	hour	station	volume
1-Oct-97	6	633	5774
1-Oct-97	6	634	5269
1-Oct-97	6	635	6191
1-Oct-97	6	636	6145
1-Oct-97	6	637	5857
1-Oct-97	6	638	6050
1-Oct-97	6	639	6040
1-Oct-97	6	641	6410
1-Oct-97	6	642	5641
1-Oct-97	6	643	1953
1-Oct-97	7	633	5456
1-Oct-97	7	634	4992
1-Oct-97	7	635	6027
1-Oct-97	7	636	6002
1-Oct-97	7	637	5806
1-Oct-97	7	638	5898
1-Oct-97	7	639	5903
1-Oct-97	7	641	6340
1-Oct-97	7	642	5564
1-Oct-97	7	643	2032
1-Oct-97	8	633	4182
1-Oct-97	8	634	4051
1-Oct-97	8	635	5106
1-Oct-97	8	636	5094
1-Oct-97	8	637	4820
1-Oct-97	8	638	5085
1-Oct-97	8	639	5401
1-Oct-97	8	641	5954
1-Oct-97	8	642	5223
1-Oct-97	8	643	1586
1-Oct-97	9	633	2866
1-Oct-97	9	634	2657
1-Oct-97	9	639	3571

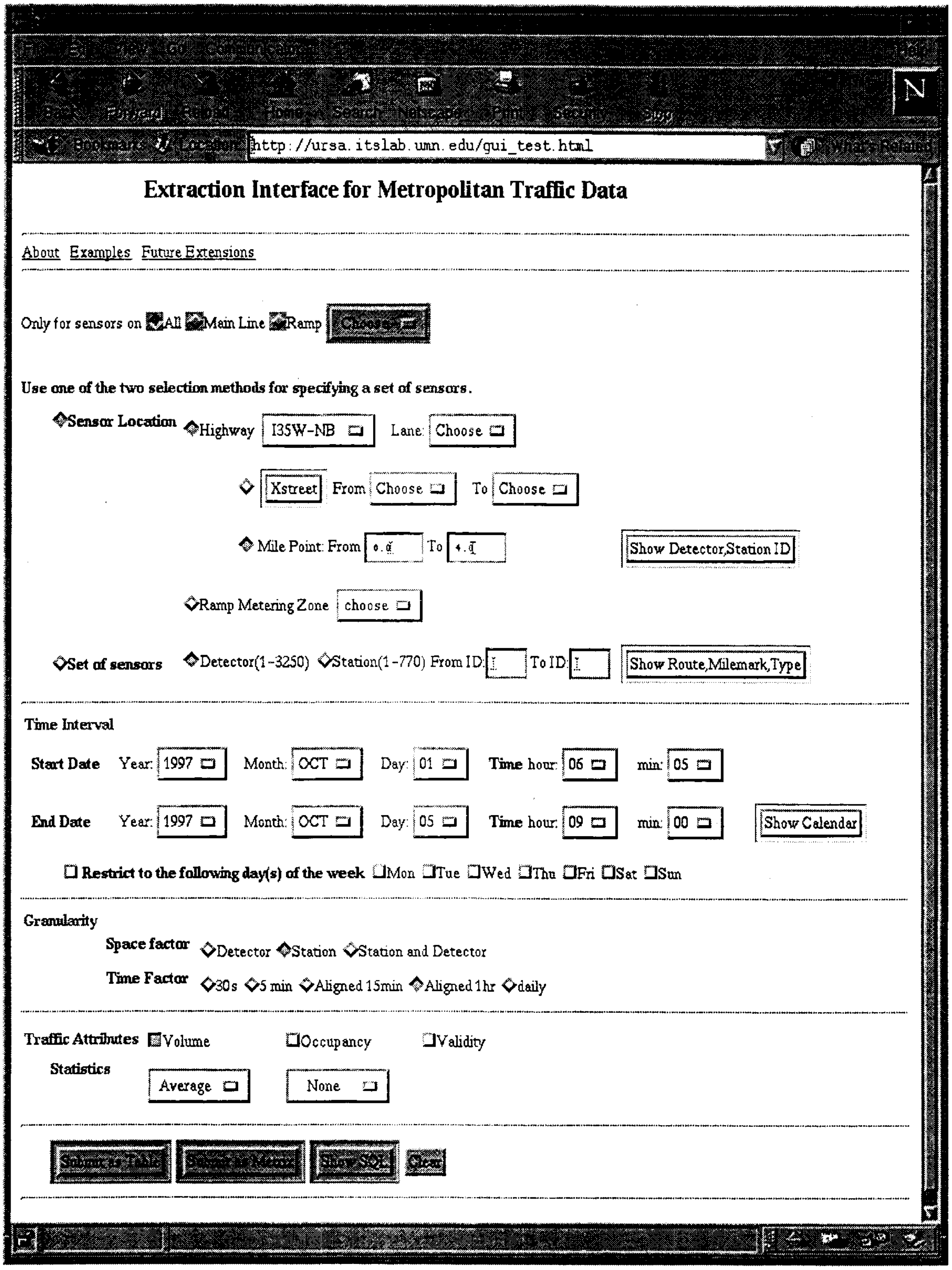

Figure 14: GUI specification for Q12

Q14. Get hourly volume for the ramp detector in zone "1A" from 6am to 9am on Oct. 2nd,

```
SELECT readdate, hour, xtan.v_hour.detector, volume
FROM xtan.v_hour, xtan.testzone
WHERE ReadDate = to_date('02-OCT-97','DD-MON-YYYY')
AND hour BETWEEN '06' AND '09'
AND testzone.zonename = '1A'
AND testzone.loctype != 'Mainline'
AND xtan.v_hour.detector = testzone.detector
```

QUERY RESULT TABLE

readdate	hour	detector	volume
2-Oct-97	6	2425	185
2-Oct-97	7	2425	294
2-Oct-97	8	2425	252
2-Oct-97	9	2425	185
2-Oct-97	6	2429	391
2-Oct-97	7	2429	350
2-Oct-97	8	2429	441
2-Oct-97	9	2429	465
2-Oct-97	6	2433	468
2-Oct-97	7	2433	478
2-Oct-97	8	2433	559
2-Oct-97	9	2433	443
2-Oct-97	6	2434	490
2-Oct-97	7	2434	599
2-Oct-97	8	2434	717
2-Oct-97	9	2434	604
2-Oct-97	6	2438	55
2-Oct-97	7	2438	72
2-Oct-97	8	2438	113
2-Oct-97	9	2438	100
2-Oct-97	6	2439	55
2-Oct-97	7	2439	98
2-Oct-97	8	2439	7
2-Oct-97	9	2439	2
2-Oct-97	6	2445	171
2-Oct-97	7	2445	190
2-Oct-97	8	2445	173
2-Oct-97	9	2445	120
2-Oct-97	6	2446	157
2-Oct-97	7	2446	174
2-Oct-97	8	2446	158
2-Oct-97	9	2446	159
2-Oct-97	6	2456	344
2-Oct-97	7	2456	320
2-Oct-97	8	2456	445
2-Oct-97	9	2456	337

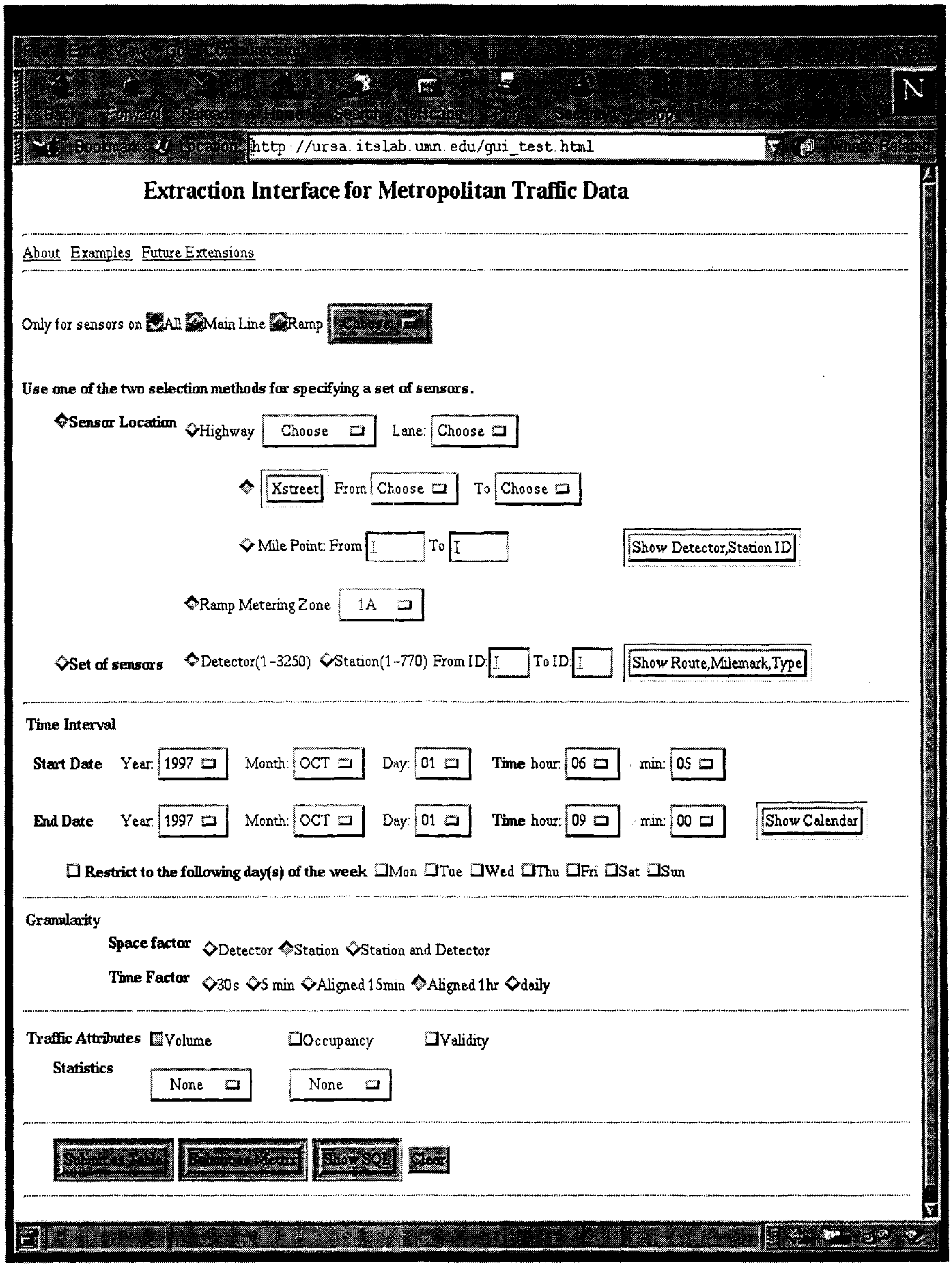

Figure 15: GUI specification for Q13

```
Q15. Get daily volume  for a set of stations on highway I35W-NB  with milepoint between 0.0 a
SELECT readdate, xtan.v_stat_daily.station, volume
FROM xtan.v_stat_daily, xtan.statrdwy
WHERE ReadDate BETWEEN to_date('01-OCT-97','DD-MON-YYYY')  AND to_date('05-OCT-97','DD-MON-
AND statrdwy.route = 'I35W-I'
AND statrdwy.mp >= 0.0
AND statrdwy.mp <= 4.0
AND xtan.v_stat_daily.station = statrdwy.station
```

QUERY RESULT TABLE

readdate	station	volume
1-Oct-97	32	33301
2-Oct-97	32	33572
3-Oct-97	32	36757
4-Oct-97	32	27875
5-Oct-97	32	18710
1-Oct-97	33	39062
2-Oct-97	33	39113
3-Oct-97	33	43084
4-Oct-97	33	33005
5-Oct-97	33	22041
1-Oct-97	35	48505
2-Oct-97	35	48505
3-Oct-97	35	52717
4-Oct-97	35	40214
5-Oct-97	35	26475
1-Oct-97	71	22515
2-Oct-97	71	22531
3-Oct-97	71	25048
4-Oct-97	71	17977
5-Oct-97	71	13200
1-Oct-97	72	36229
2-Oct-97	72	36210
3-Oct-97	72	39968
4-Oct-97	72	30611
5-Oct-97	72	20582

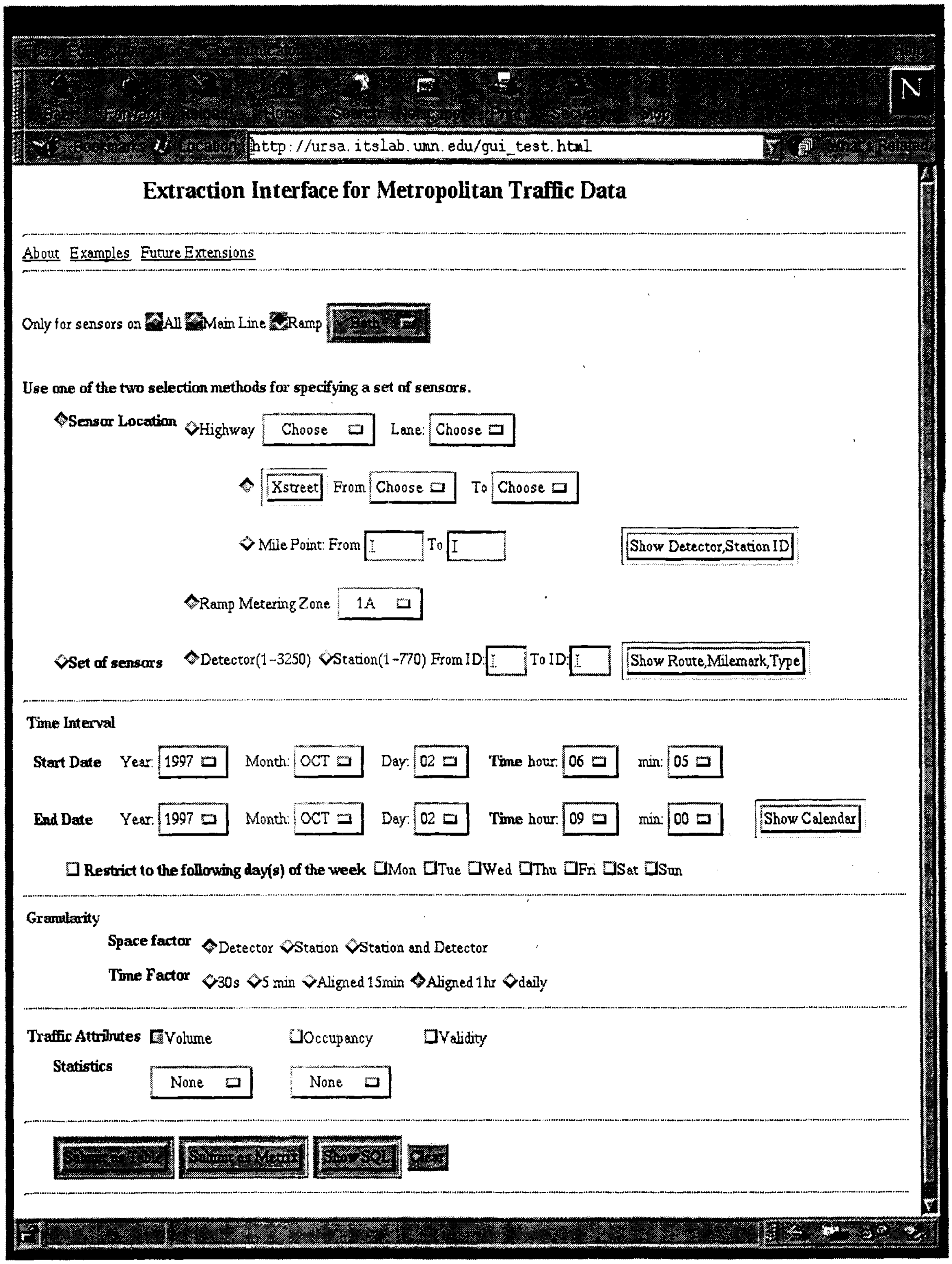

Figure 16: GUI specification for Q14

Q16. Get the AM/PM peak hour, peak hour volume for station ID = 5 on Oct. 1st, 1997

Q17. Get the AM congestion hours for a set of stations on highway I35W on Oct. 1st, 1997

Q18. Get the count of valid readings for all detectors on Oct. 1st, 1997

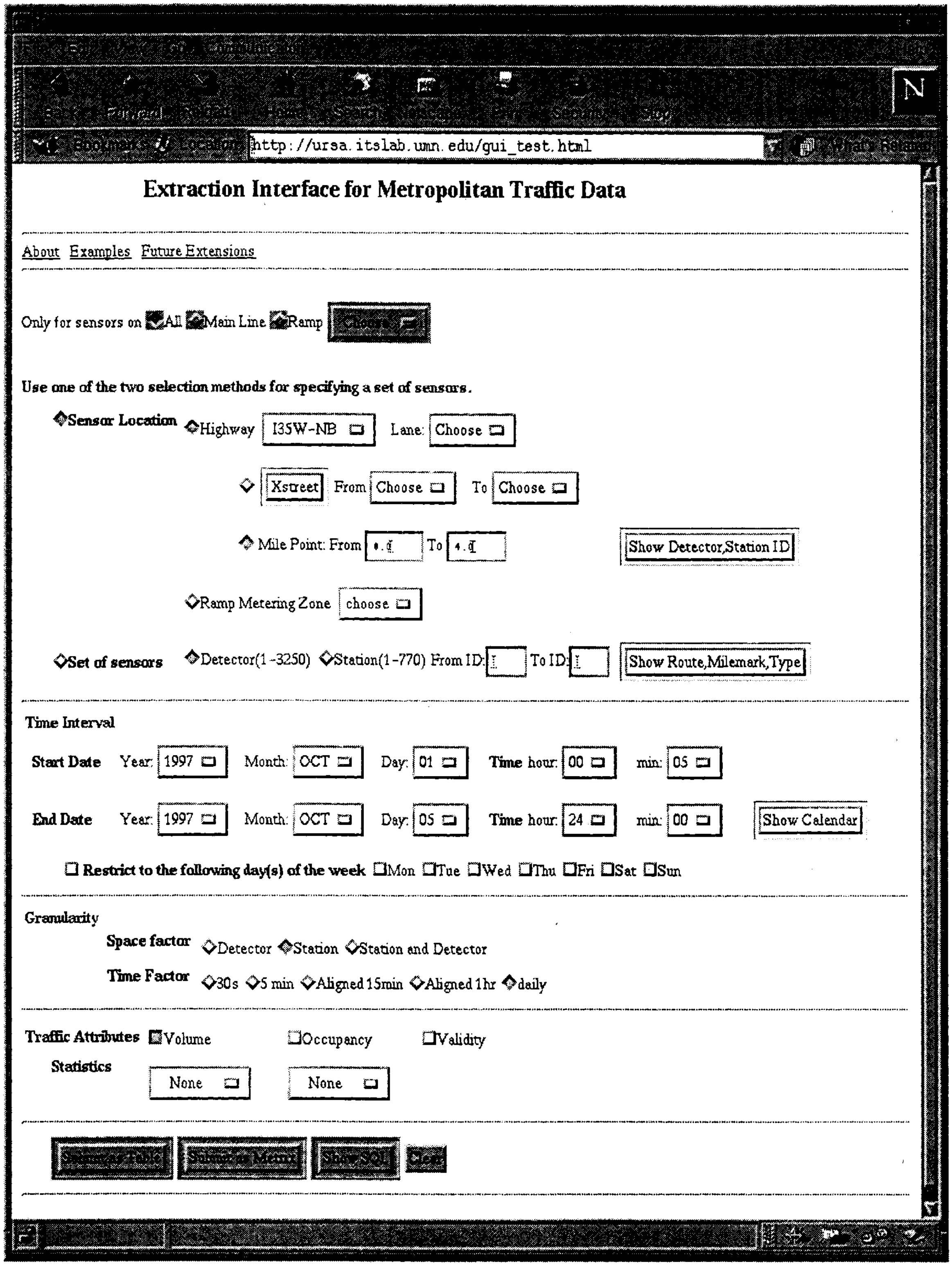

Figure 17: GUI specification for Q15