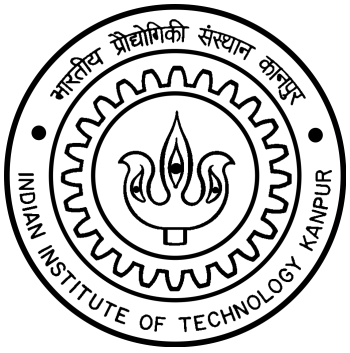
DBProjector : A Web-Based Tool for Querying, Analysis and Visualization of Data

A thesis submitted
in Partial Fulfillment of the Requirements
for the Degree of

Master of Technology

by

Swapnil Sopan Mahajan
Roll Number: **13111028**



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

May, 2016

CERTIFICATE

It is certified that the work contained in the thesis titled **DBProjector : A Web-Based Tool for Querying, Analysis and Visualization of Data**, by **Swapnil Sopan Mahajan**, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Dr. Arnab Bhattacharya

Department of Computer Science and Engineering

IIT Kanpur

Dr. Amey Karkare

Department of Computer Science and Engineering

IIT Kanpur

May, 2016

ABSTRACT

With the growing trend of data analytics, the needs for analysis and visualization of data has become paramount. Data analytics involves examining data in ways that reveal the relationships, patterns, trends, etc. that can be found within it. These relationships, patterns, trends, etc. lead to accurate assessment in order to better understand the actions and their impacts. However, it is difficult to make sense out of raw data just by looking at in its natural form. Adding easy querying and visualization capabilities makes it easily understandable to everyone.

Another emerging trend is the availablitiy of usage of large databases over the web using the browser in both traditional and mobile forms.

In this thesis, we have designed and developed a web-based tool DBProjector that acts as both data browser as well as data visualizer and helps in data analytics. If a database is available over the web, the source URL can be plugged to the tool to allow its querying, analysis and visualization. We have used this tool to perform case studies on ITS (Intelligent Tutoring System) and NAQI (National Air Quality Index) to showcase its usefulness.

Dedicated to all who inspire me

Acknowledgements

First and foremost of all I would like to express immense gratitude to Prof. Arnab Bhattacharya and Prof. Amey Karkare, my guide and my co-guide for their constant motivation, inspiration, guidance and support. Their expertise & experience in the subject and their timely & accurate guidance are the only reason that I have been able to complete my thesis work.

On every single day spent at IIT Kanpur, I felt lucky to be a part of this institute. No amount of gratitude is enough and I will always be in debt to the professors of IITK. Not to mention the world class state-of-art infrastructure provided by the institute and maintained by the staff. I am grateful them.

Thanks to PhD scholars Tejas Gandhi and Shahbaz Khan for theirs suggestions. I would also like to mention my friends who made me do things which I now feel good of, who helped practice my sarcasm and have been constant motivation for the same.

Contents

List of Figures

Chapter 1

Introduction

Analyzing the data has always offered great benefits to organizations of all sizes and across all industries. It involves examining data in ways that reveal the relationships, patterns, trends, etc. that can be found within it. These relationships, patterns, trends, etc. lead to accurate assessment in order to better understand the actions and their impacts. It also leads to understanding the behavioral patterns of the users which can help in decision making of different facets of the system including architecture, deployment, etc.

But raw data is boring and it's difficult to make sense out of it just by looking at in its natural form. Adding visualization to it make easily understandable by everyone. Visualization will also make it faster to make sense out of it and also interesting patterns can be observed that would not be apparent from looking only at the raw data.

There are great tools available to browse the data in the raw form. There are also great charting libraries available to present the given data in best way possible to make it easily understandable. The problem that we face here is that both these areas have been addressed independently. Hence for one to extract data and plot it in most comprehensible way, he has to write scripts to dump data from database browsing applications and use charting libraries to plot it. Sometimes he does this manually and he has to tweak the scripts to try out different ways of visualizing the

data to select the best one.

In this thesis we are trying to address this area and we have designed and developed DBProjector which acts as both data browser as well as data visualizer. First of all it is a web-based tool so requires no local installations and can be accessed by anywhere. And the DB server to be analyzed need not to be on the same machine where DBProjector is hosted as far as the DB server allows connections from that machine. You can also save the queries so that other people who have access to the DB can view results of queries and their respective charts.

Chapter 2

Background and Related Work

This section will present some of the tools available which can perform browsing of the data or plotting charts of such data.

2.1 Data querying tools

2.1.1 MySQL Workbench

MySQL Workbench[1] is a visual tool distributed as desktop application. It includes features like DB designing using the schema designer, performance dashboard to track performance analytics of the DB and query tool for data browsing. This tool supports operating systems: Windows, Linux and Mac OS X and is distributed under the license GPL. It however does not have data visualization capabilities. This tool mainly gives visual aid over MySQL CLI.

2.1.2 Database Workbench

Database Workbench[2] is a similar tool to the above one except this has few extra features like debugging triggers and procedures, comparing data among different databases, shared workspace for teams. This is a proprietary tool and is distributed as desktop application supporting Windows, Linux and Mac OS X. This tool also lacks the feature of visualizing the database.

2.2 Data visualization tools

2.2.1 Data Driven Documents

D3js[3] is the most popular visualization library. The reason of it being most popular is that it provides very detailed and low level APIs to create SVGs. This makes it possible to create any chart that one can think of. This is distributed under BSD license. But it does not ship with pre-built charts out of the box. But there is nice gallery which showcases what is possible with it. For common charts using this library is a overkill. This is just charting library and requires data to be passed in given format.

2.2.2 FusionCharts

FusionCharts[4] is again another javascript charting library but not open-source as that of D3js. It comes with few pre-built charts among which some have advanced features like zooming, linked charts, data filtering. Being a proprietary product this library has well written API docs and examples which makes it easy to learn.

All the above applications/tools perform either of querying the data or visualizing the tailored data. There has not been done more work on creating application mainly web-based(so that it becomes OS independent) which does both the parts. This is the main intention of this thesis.

Chapter 3

Case Studies

3.1 Case Study 1: Intelligent Tutoring System

Using the DBProjector , a case study has been performed on Intelligent Tutoring System(ITS)[5] and the results of the study as presented in this section.

3.1.1 Introduction

ITS is an web based education platform that conducts courses online. It was developed by my colleague Rajdeep Das at IIT Kanpur as his M.Tech. thesis under the supervision of Prof. Amey Karkare in the department of Computer Science and Engineering. It aimed to provide enhanced introductory programming learning experience to the masses by the means of technology and at the same time to be used as a platform for data collection and analysis.

The database used for the analysis is from the semester July-November'2015.

3.1.2 Schema

Figure 3.1 represents a minified schema of the Intelligent Tutoring System (ITS). It shows only the important columns from the tables. Figure 3.2 shows the expanded view of the tables related to compilation table.

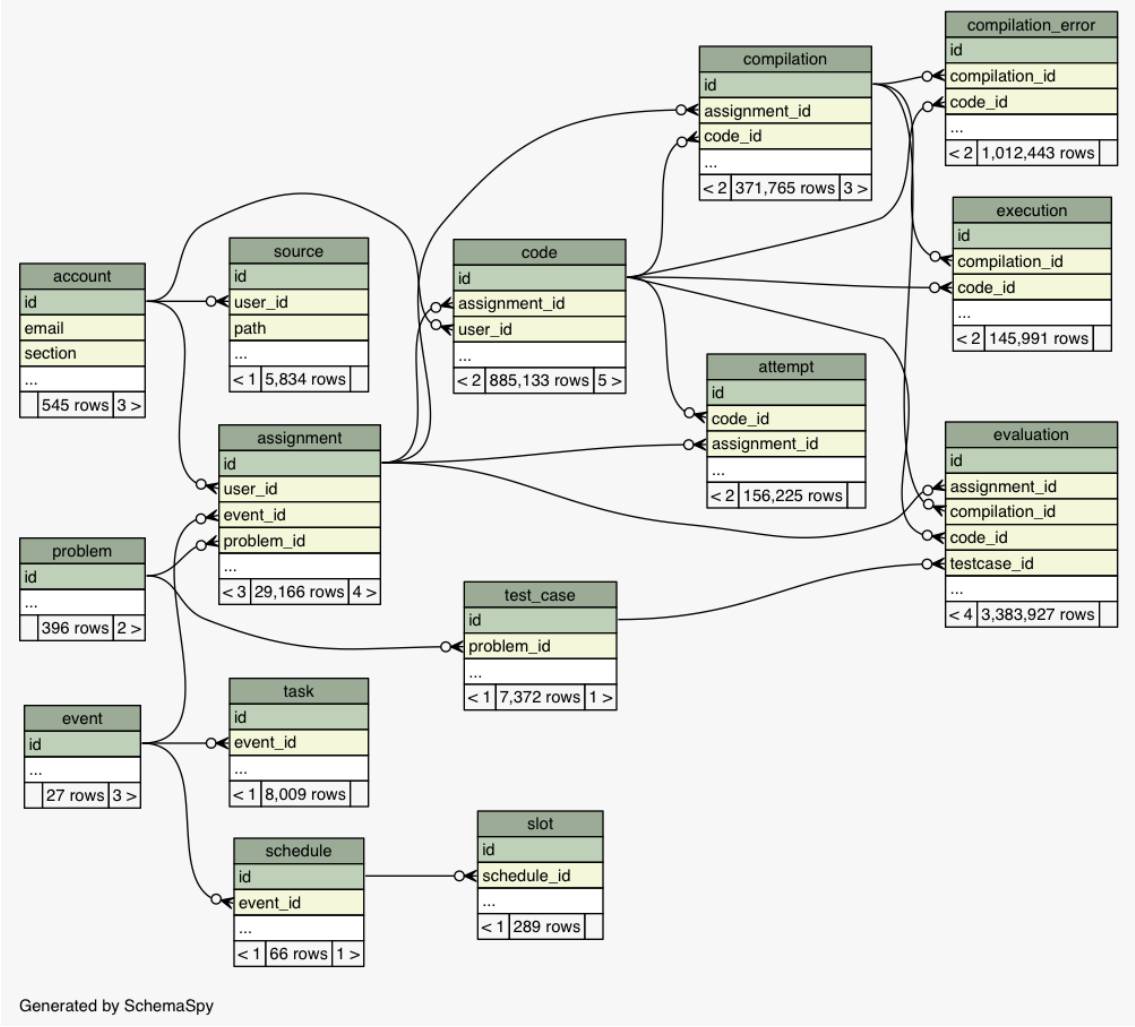


Figure 3.1: ITS Schema

Figure 3.2 shows that every compilation is associated with an assignment and corresponding code. On every compilation, compilation errors, if any, are added to the compilation_error table. The relationships with evaluation and execution table suggests that on every evaluation and execution, compilation is first triggered for respective code.

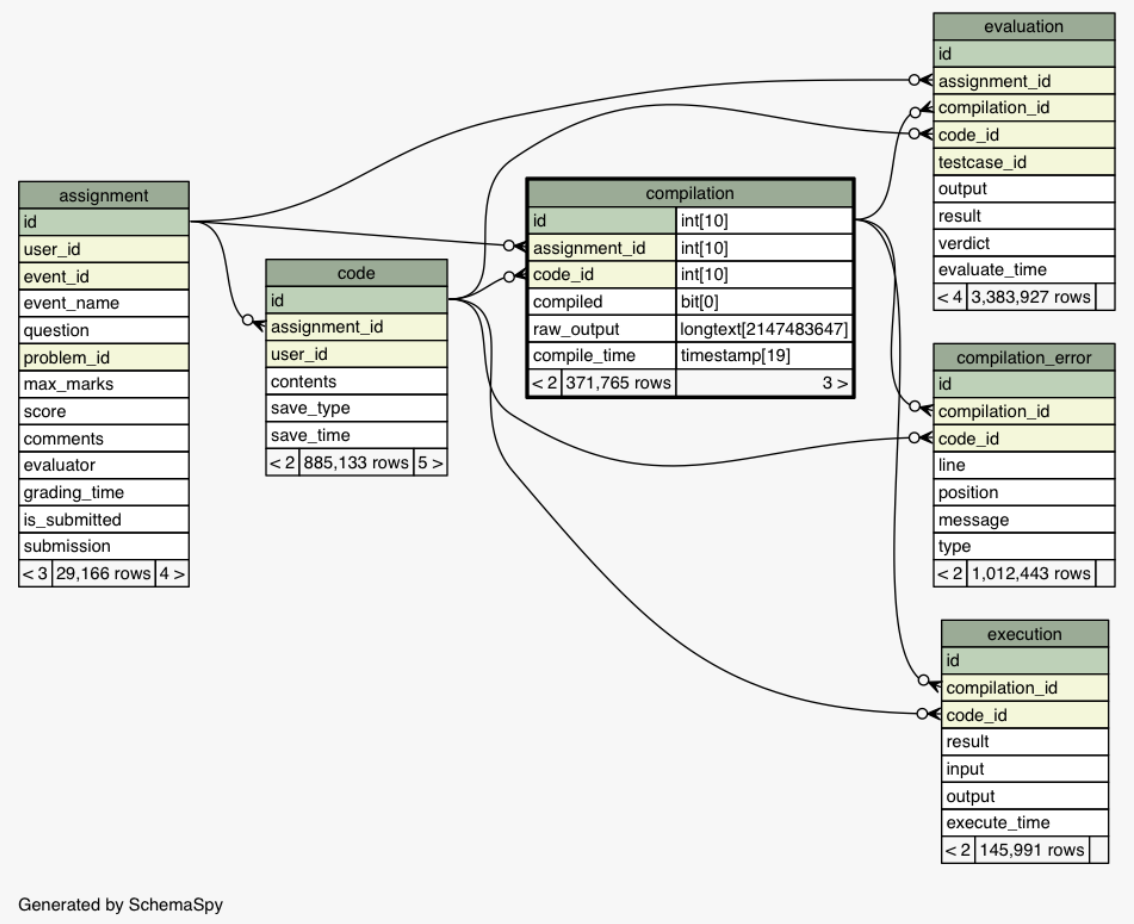


Figure 3.2: Compilation Table Relationships

3.1.3 Data Visualization and Analytics

Compilations

Figure 3.3 shows the comparison of successful vs unsuccessful compilations over all labs. It shows that among all the compilations around 20% are unsuccessful and resulted in compilation error.

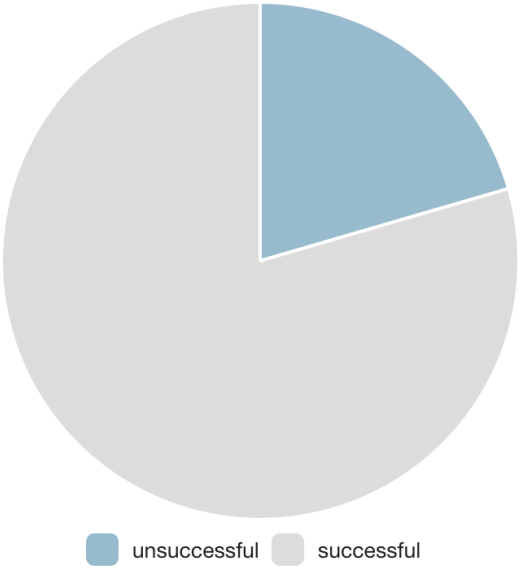


Figure 3.3: Successful vs Unsuccessful Compilations

Figure 3.4 shows total compilations per lab. Total compilations seem to reduce gradually after the second lab except seventh lab as the students get used to the platform and get acquainted with coding and syntax. Reasons behind lab seven having highest number of compilations could be tricky corner cases or difficult problem statements.

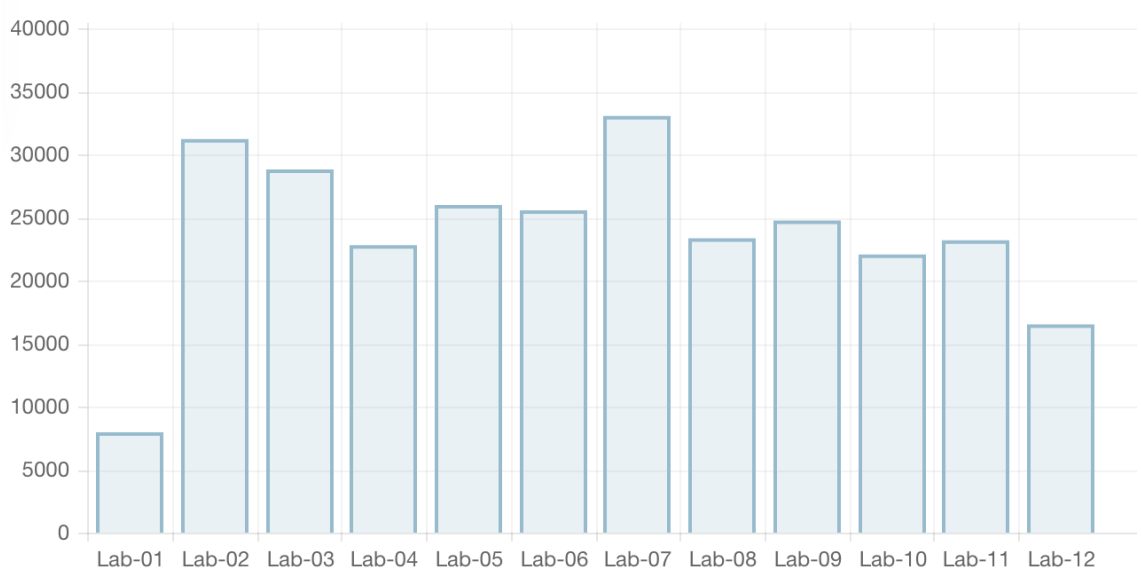


Figure 3.4: Compilations per lab

Evaluations

Figure 3.5 and Figure 3.6 shows total evaluations and failed evaluations per lab. Significantly higher number of evaluations in seventh lab supports our claim of having tricky test cases.

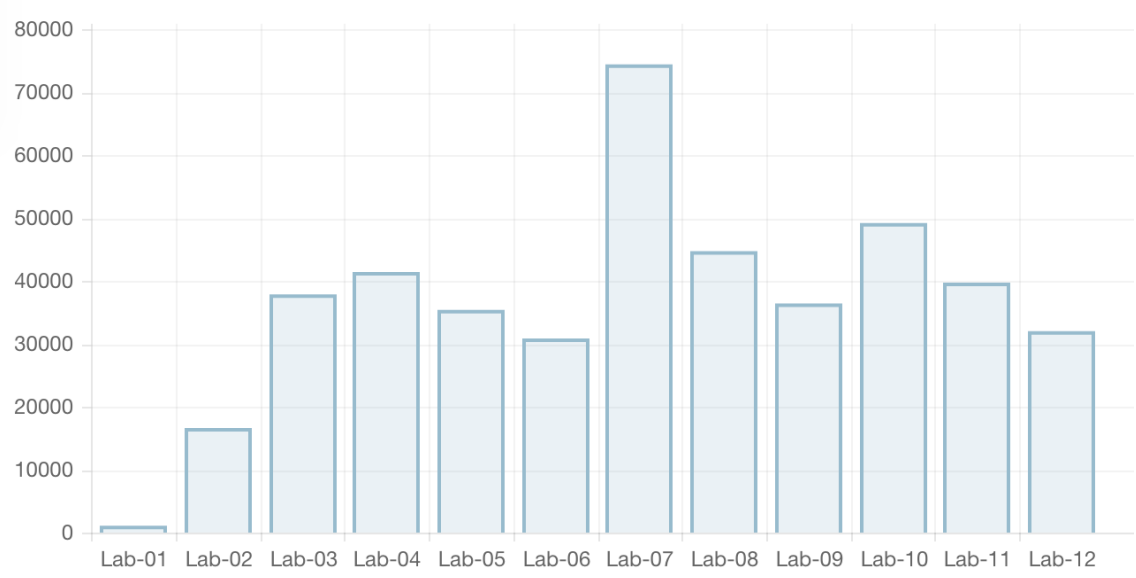


Figure 3.5: Evaluations per lab

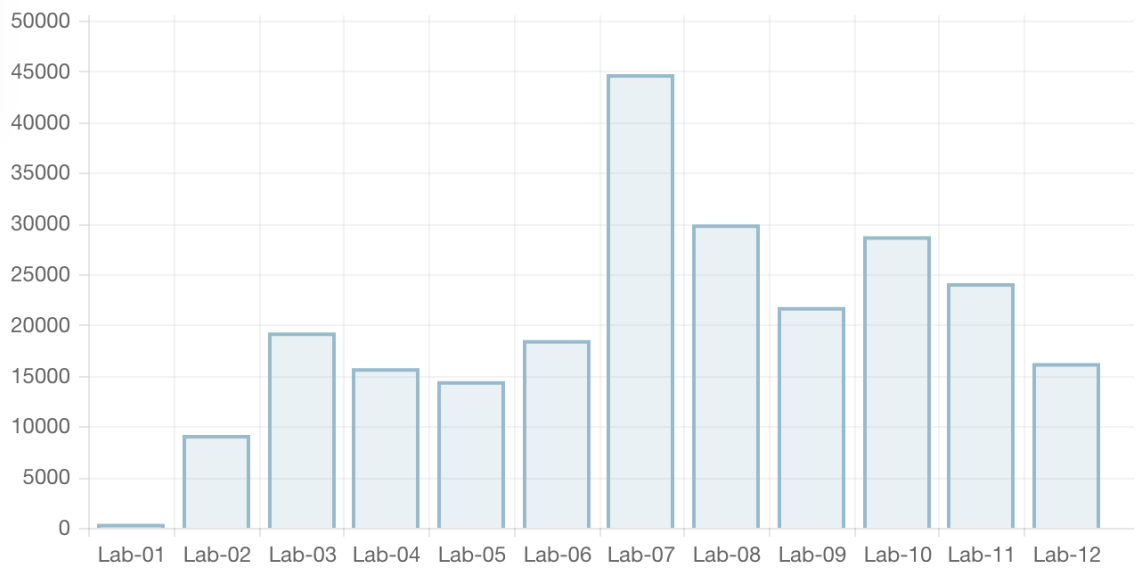


Figure 3.6: Failed evaluations per lab

Marks

Figure 3.7 and Figure 3.8 shows the marks distribution for lab 7 and lab 10 respectively. In spite of many evaluations in Lab 07, many students have got full marks as compared to that of Lab 10. While many have scored 0 in Lab 10 as compared to Lab 07.

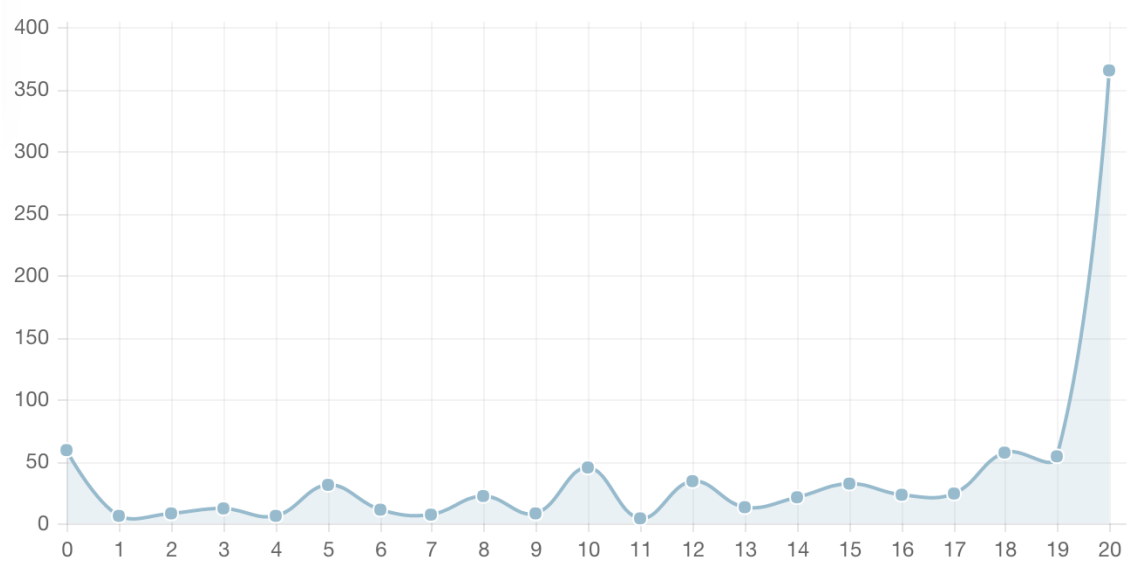


Figure 3.7: Marks distribution for Lab 07

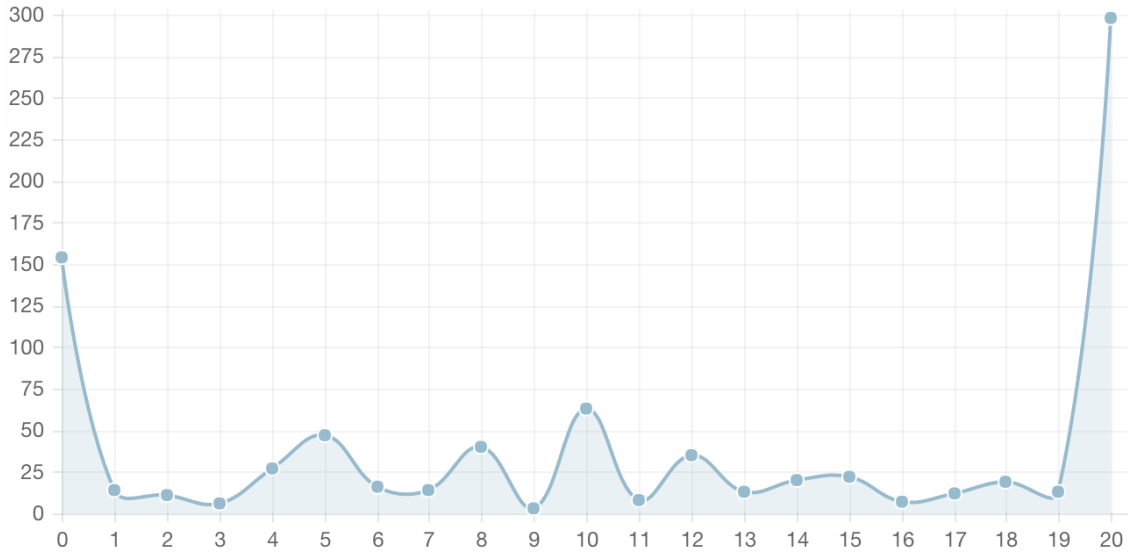


Figure 3.8: Marks distribution for Lab 10

3.2 Case Study 2: National Air Quality Index

In this section the results of the case study performed on National Air Quality Index have been presented.

3.2.1 Introduction

National Air Quality Index[6] is air quality monitoring and dissemination system. Awareness of daily levels of air pollution is important to the citizens, especially for those who suffer from illnesses caused by exposure to air pollution. This system helps such people by displaying the real time air quality statistics.

3.2.2 Data Visualization and Analytics

Stations

Figure 3.9 shows the average $PM_{2.5}$[7] values at different stations in Delhi. It shows that Anand Vihar ranks first while Shadipur and R. K. Puram stand second and third when it comes to $PM_{2.5}$. $PM_{2.5}$ is the most prominent pollutant in Indian cities mainly due to the weather and increasing pollution due to industries and vehicles.

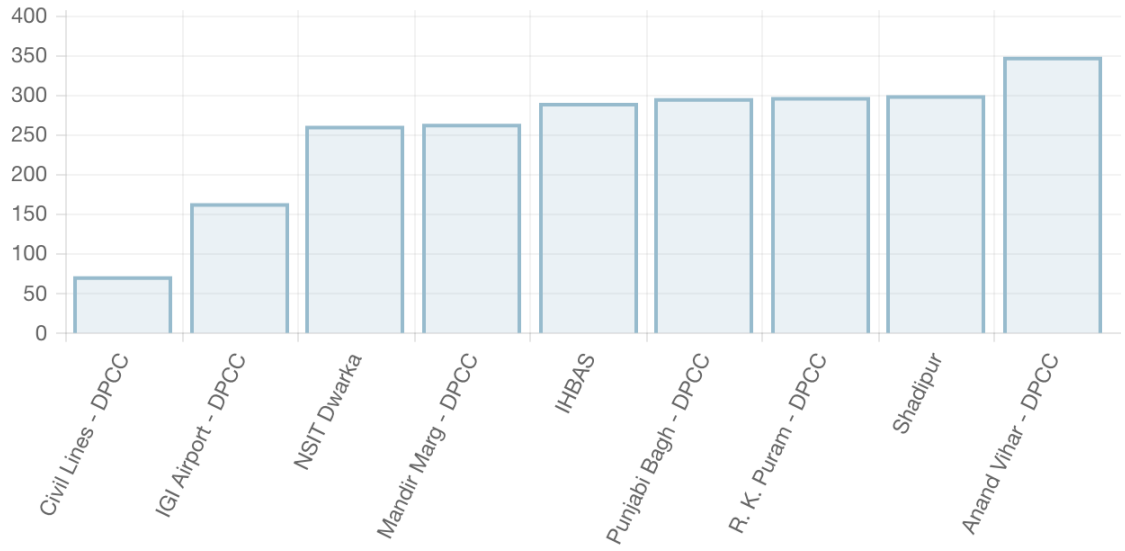


Figure 3.9: Average $PM_{2.5}$

$PM_{2.5}$ trend

Figure 3.10 shows the trend of $PM_{2.5}$ levels during a typical day. It shows that the levels are high during 9AM to 12AM and after 6PM which is the usual time of high traffic. Lower levels during 12AM to 5PM point towards the effect due to temperature change and lower traffic.

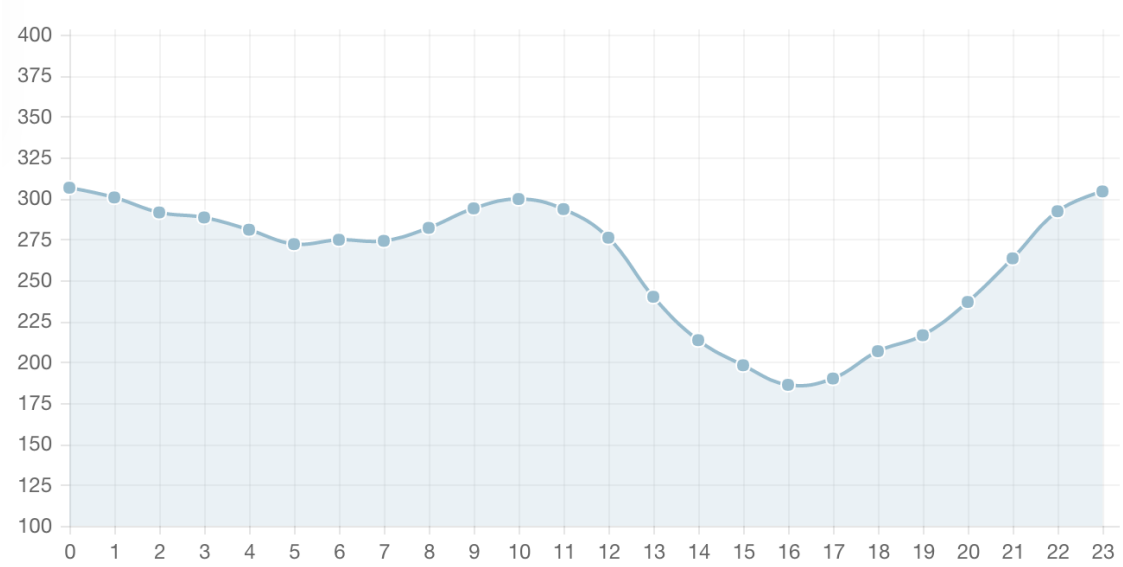


Figure 3.10: $PM_{2.5}$ trend during a day

Odd-Even Policy

Odd-Even policy was applied in Delhi during January 1 and January 15. Under this policy vehicles with odd and even numbers were allowed only on odd and even dates respectively. The intention of this policy was to reduce the number of vehicles on the roads which in turn was supposed to reduce the $PM_{2.5}$ levels. The analysis of the data gathered just before and after this policy has been presented in Figures 3.11 and 3.12.

Figure 3.11 shows the average $PM_{2.5}$ levels during last 15 days of December 2015 and first 15 days of January 2016. The levels are seen to be increased in January. This change is due to the climate changes as the similar changes in the levels have been seen in the first 15 and last 15 days of January in Figure 3.12.

Hence from these statistics the Odd-Even policy does not seem to have helped in

reducing the pollution.

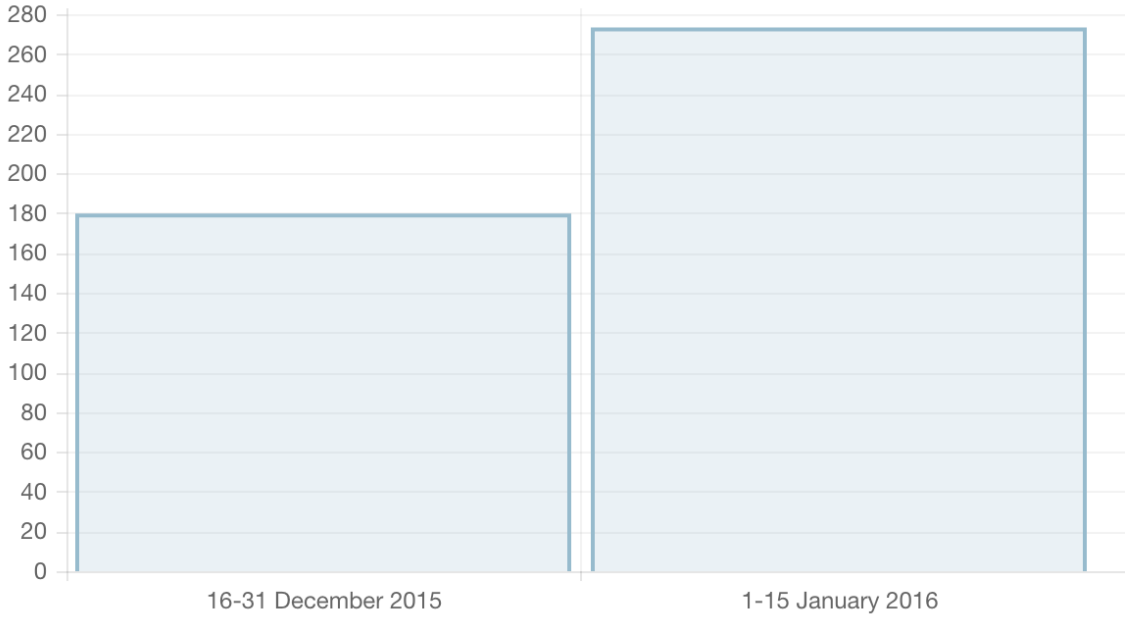


Figure 3.11: $PM_{2.5}$ during December 2015 and January 2016

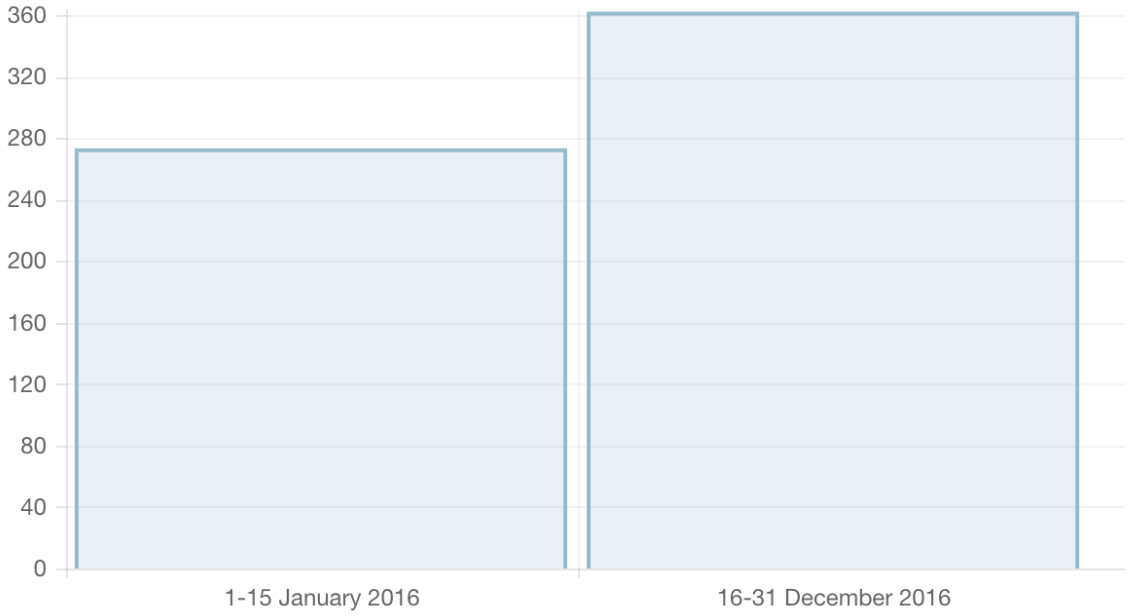


Figure 3.12: $PM_{2.5}$ during days of January 2016

Chapter 4

Design and Implementation

4.1 Components

This section represents the different components present in the system and their relation with the rest of components.

This system is developed using Client-Server architecture. Hence there are mainly three components in it viz. Frontend, Backend and External DB server

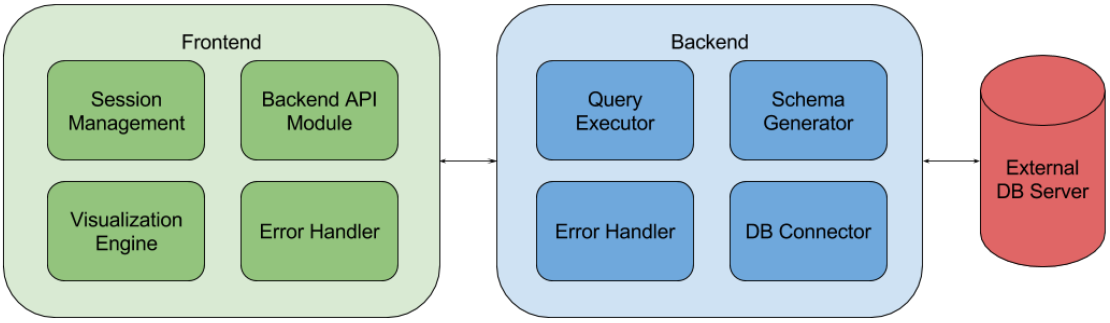


Figure 4.1: Components

4.1.1 Session Management

This module is responsible for holding the information needed for a session such as DB host, DB user, DB password, etc. Once a new connection is made, this module store the above said information. The information stored is in the scope of tab which is open. Once the tab/browser is closed, this information is lost and a new

connection is made if user tries to revisit the same web page.

4.1.2 Backend API Client

There is a clear separation of frontend and backend functionality. Backend provides the RESTful[8] APIs so that anyone can build their own frontend or can directly consume the data without having any frontend.

This module is responsible for converting the requests in such a format that the backend can understand. It also knows about the type of response backend is going to generate. It parses the backend response, adds new fields to it if needed(e.g in case of errors) and then passes the response to callback function provided.

4.1.3 Visualization Engine

This module is responsible for plotting the charts. It takes data, chart type and axes as the input for plotting these charts. Once the chart type and axes are selected, this modules extracts the relevant information from the data in the form of list for the axes and the passes it to a chart drawing library.

4.1.4 DB Connector

This module is responsible selecting the right driver for the specified database connection and then connecting to that DB for further communication. Frontend calls the API exposed by this module to check if the database parameters such as host, user, password, database name are correct and the user has read access to the DB.

4.1.5 Query Executor

This module is responsible for invoking the DB connector module and passing the query in right format to for execution. After execution it takes the result set and parses it. This parsing involves extracting the column names from the result set

which can later be used by the Visualization Engine as candidate columns for the axes of charts.

4.1.6 Schema Generator

This module is responsible for invoking the external tool SchemaSpy[9] for retrieving the schema of database and storing it HTML pages which can later be viewed by the user. Once the DB connector module passes the DB credential check, this module is invoked.

4.1.7 Error Handler

Both frontend and backend have their respective error handlers which take care of capturing the error and presenting it to the user/api caller in more unified way.

The backend error handler capture errors like connection failures, invalid query syntax, external tool invocation errors, etc. While fronend error handler deals with capturing the backend connection failures, backend api failures and then presents these problems to user with required information at relevant places.

4.2 Component Interaction Model

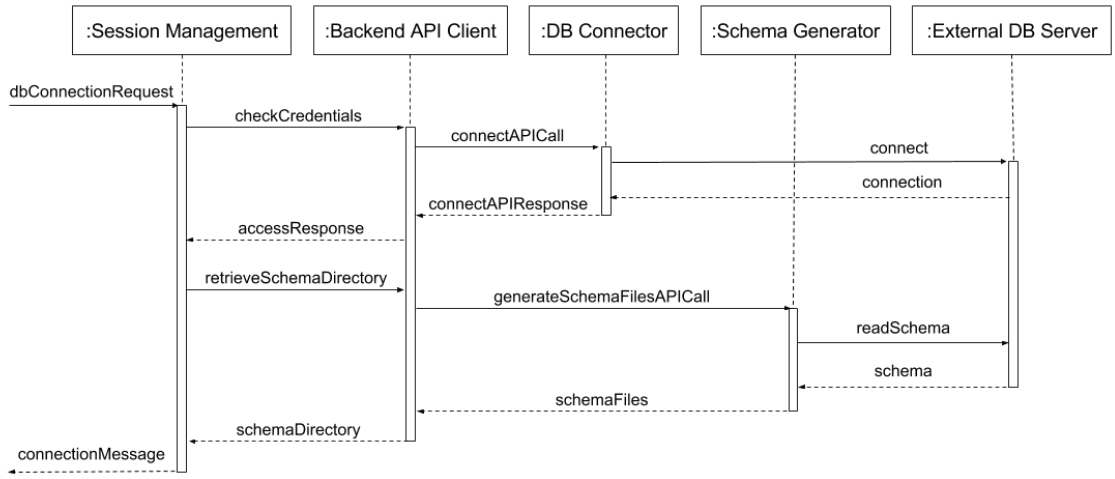


Figure 4.2: Sequence Diagram for DB Change Request

The above diagram represents the flow of database connection change request by the user. The DB connection parameters are first saved in a temporary scope. Then Backend API client method is invoked by passing the connection parameters and callback functions for successful and unsuccessful scenarios. Then API call is made to the backend which tries to create a connection with the external DB server. On correct connection parameters, external DB server returns a connection. After this successful call the DB connector returns response with status as success.

Once the connection parameters are verified to be correct, session management module transfers the parameters stored in temporary scope to main scope and triggers schema retrieval API. The API client sends a well-formatted http request to schema generator. Schema generator then reads the schema from external DB server using the credentials and stores this schema in files in html format which can later be served on demand. Once the html files are created, schema generator returns the directory where the schema files are stored to the API client which passes it to session management module. Session management module stores the schema location in main scope for the current session.

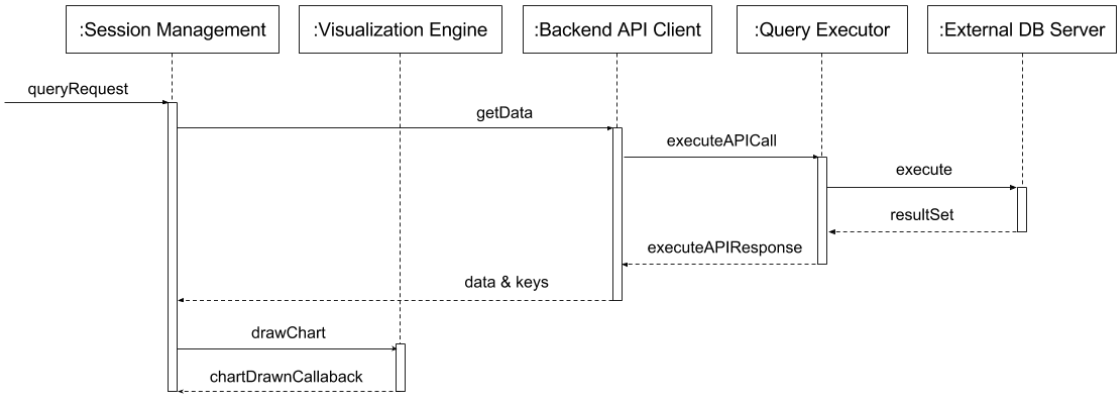


Figure 4.3: Sequence Diagram for Query Execution and Plot Chart Action

The above diagram represents the flow of custom query being executed and the data being plotted in the form of chart selected. The session management module sends the locally stored database connection parameters to backend API along with the query to be executed. Query executor then creates connection with the external DB server and executes the query. The resultset is then fetched from DB server. Query executor then extracts the columns from the query and attaches those to the response along with the resultset. Once the data and keys are available, session management module sends this information to the visualization engine along with the type of chart and the axes selected by the user. This is an asynchronous call. Visualization engine then creates a canvas and plots the chart with given data.

4.3 Technology stack

4.3.1 Frontend

AngularJS[10]

The base of the frontend component is mainly AngularJS. Because of the two-way data binding feature of AngularJS, it has reduced a lot code else needed to display and read data from html elements into javascript respectively.

Angular Material[11]

Angular Material has been used as the UI framework. It provides simpler syntax and boilerplate to create UI elements which are compliant with Google's Material Design guidelines. It provides uniform experience over different sized devices likes desktops, phones, etc.

ChartJS[12]

ChartJS is the javascript library used for plotting the charts. This library uses the newer HTML5 canvas APIs to draw charts instead of using the old fashioned SVGs, which makes if faster and compatible with most of the devices. ChartsJS library also has it's angular adapter available which makes it even easier to integrate it with the data. It is as easy as assigning the values to variables and ChartJS will automatically plot the chart for you.

4.3.2 Backend

Flask[13]

Flask webframework(often described as micro framework) has been used for the backend. As the frontend is a single page angular app, the backend needed only to provide the RESTful APIs. Hence we have chosen this lightweight webframework.

SchemaSpy

SchemaSpy is a java based tool which analyses the metadata of the schema in a database and generates a visual representation of it in a browser-displayable format. The system provides it the connection parameters so that it can pull the metadata of the external DB server.

Chapter 5

Conclusions and Future Scope

5.1 Conclusions

In this thesis we have designed and developed a web-based system to query, visualize and analyze from any database server accessible from the deployment machine. We have tried to keep the interface simplistic and intuitive for ease of use.

Instructors can use this system to analyze the trends and patterns in examinations. Students can use this system to analyze data they have to extract out useful information from it. Even the system administrators can analyze their system based on the logged data they have.

5.2 Future Work

5.2.1 Support for different types of Databases

Currently DBProjector supports only MySQL compliant databases. We can add support for other relational as well as NoSQL databases. This will make it a one stop solution for all types of DB data analysis.

5.2.2 More visualization chart types

Addition of more visualization chart types like map chart, scatter chart will make the system more usable and apt for certain datasets. Addition of such charts will also result in more dimensions.

References

[1] MySQL. *MySQL Workbench*. URL: https://www.mysql.com/products/workbench/.

[2] Upcene. *Database Workbench*. URL: http://www.upscene.com/database_workbench/.

[3] Mike Bostock. *Data Driven Documents*. URL: https://d3js.org/.

[4] InfoSoft Global. *FusionCharts*. URL: http://www.fusioncharts.com/.

[5] Rajdeep Das. "A Platform for Data Analysis and Tutoring For Introductory Programming". M.Tech. thesis. India: Indian Institute of Technology Kanpur, 2015.

[6] CPCB. *National Air Quality Index*. URL: http://164.100.160.234:9000/.

[7] Wikipedia. *Particulates*. URL: https://en.wikipedia.org/wiki/Particulates.

[8] *Representational state transfer*. URL: https://en.wikipedia.org/wiki/Representational_state_transfer.

[9] *SchemaSpy*. URL: http://schemaspy.sourceforge.net/.

[10] *AngularJS*. URL: https://angularjs.org/.

[11] *Angular Material*. URL: https://material.angularjs.org/.

[12] *ChartJS*. URL: http://www.chartjs.org/.

[13] *Flask*. URL: http://flask.pocoo.org/.