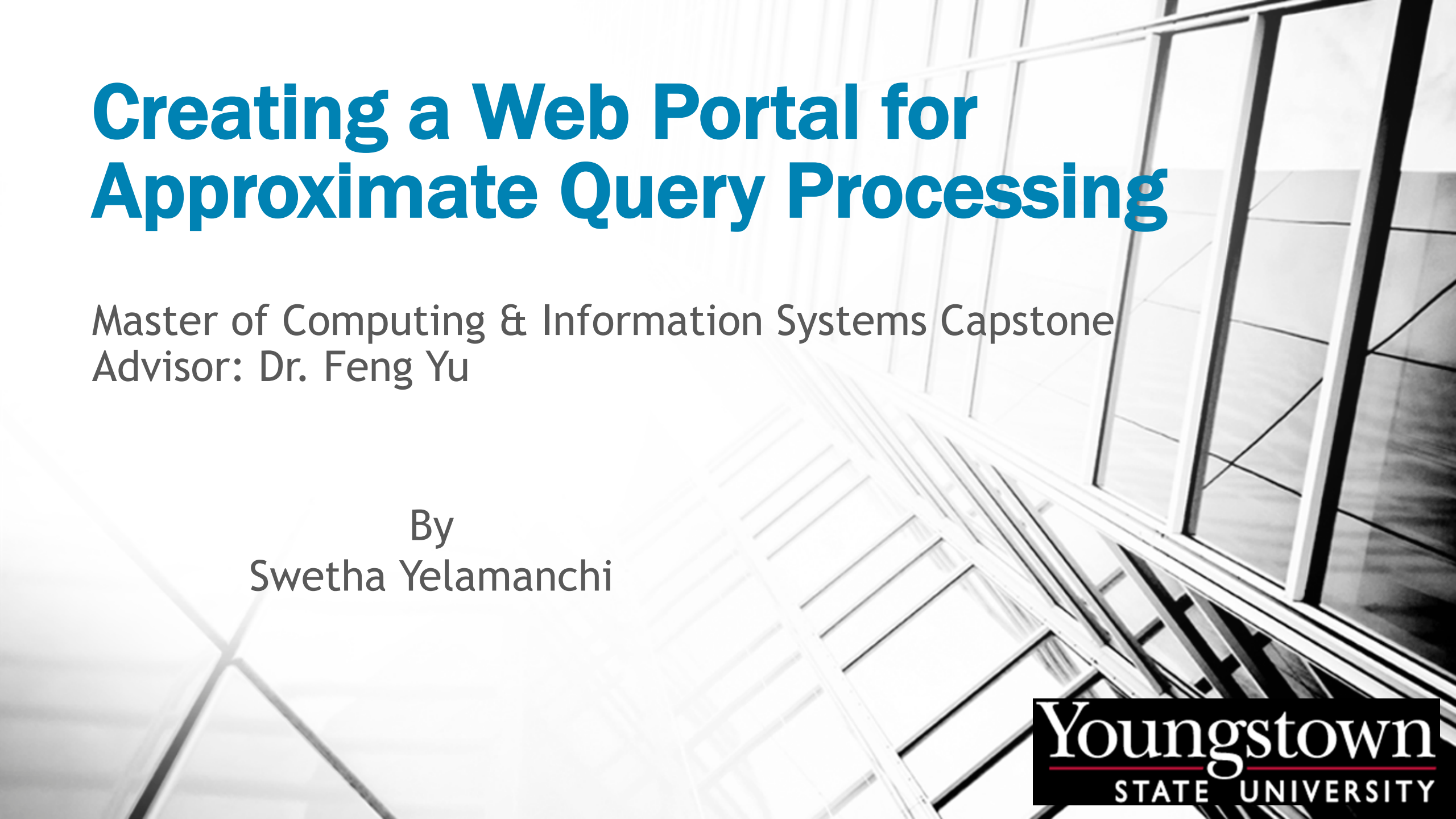
Creating a Web Portal for Approximate Query Processing

Master of Computing & Information Systems Capstone
Advisor: Dr. Feng Yu

By
Swetha Yelamanchi



Agenda

1. Introduction
2. Literature review
3. Project Flow
4. Results and Analysis
5. Conclusion

Introduction

- A server-side web application was created using node.js express framework and embedded java script.
- The front-end web interface shows the text editor to submit a SQL query.
- Once submitted the application displays performance metrics and results of both Postgre and sample Postgre Databases.

Literature review

Node.js

- Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications.
- Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.
- Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications which are written in JavaScript.
- Node.js = Runtime Environment + JavaScript Library

Features of Node js

- **Asynchronous and Event Driven** - All APIs of Node.js library are asynchronous, that is, non-blocking
- **Very Fast** – Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution

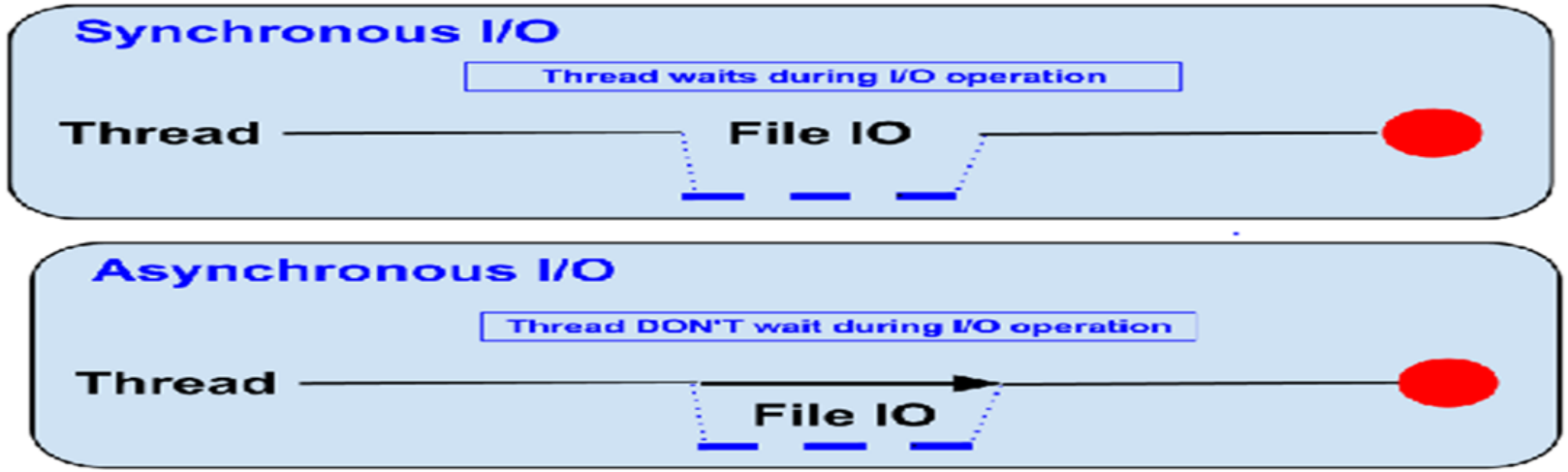


Fig 1: Asynchronous and Event Driven

Features contd…

- **No Buffering** – Node.js applications never buffer any data. These applications simply output the data in chunks
- **Single Threaded but Highly Scalable** – Node.js uses a single threaded model with event looping.

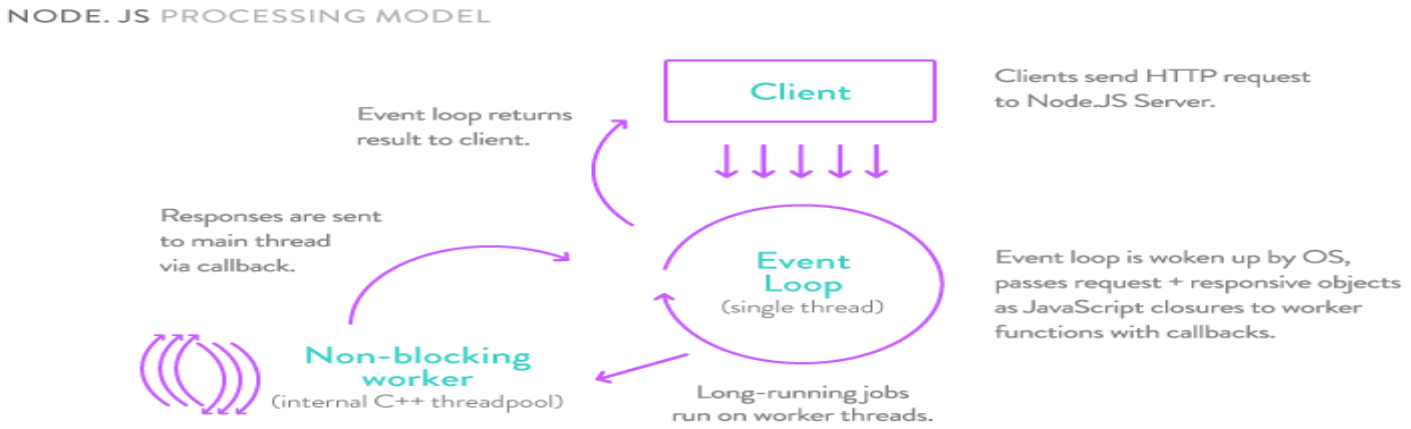


Fig 2: Single Threaded but Highly Scalable

Node.js - Express Framework

- Express is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications. It facilitates the rapid development of Node based Web applications.
- Allows to set up middlewares to respond to HTTP Requests.

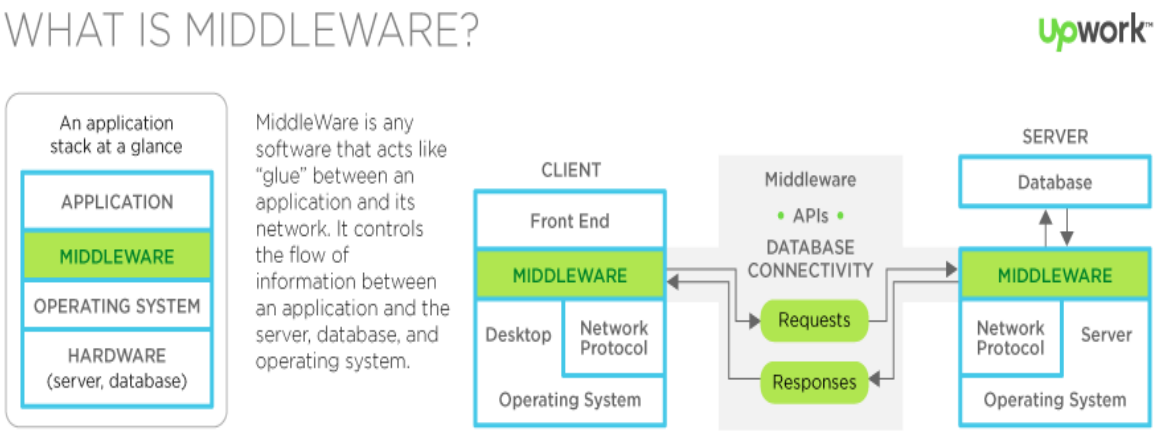


Fig 3: Middleware

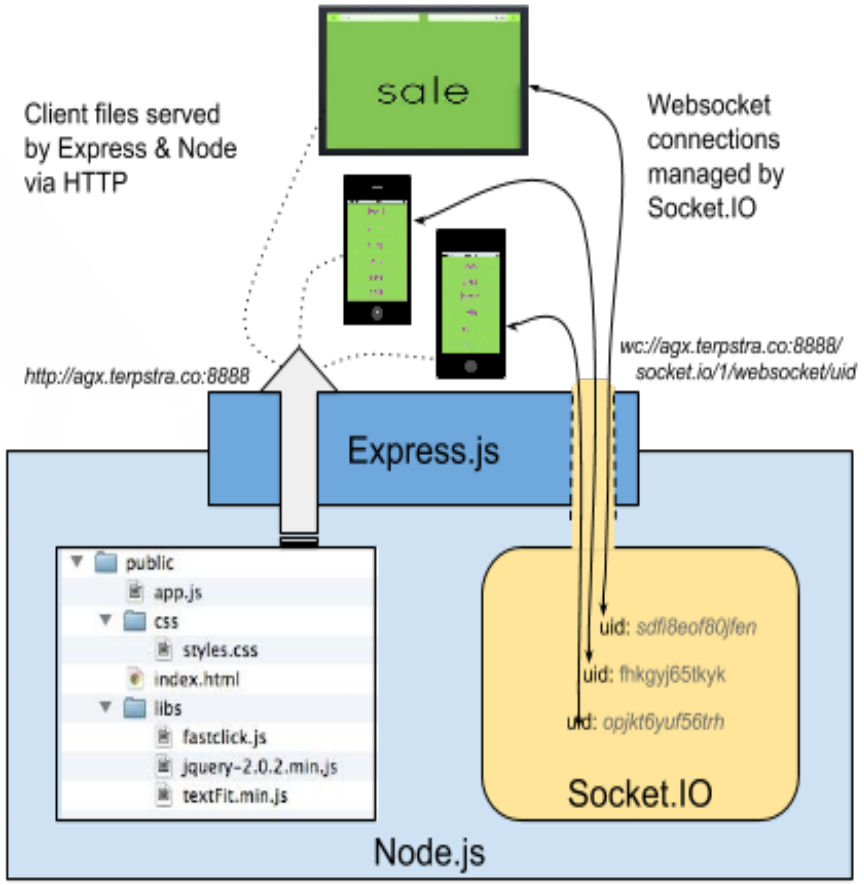


Fig 4: Node.js - Express Framework

NPM (Node Package Manager):

- NPM is a package manager for Node.js packages, or modules if you like.
- www.npmjs.com hosts thousands of free packages to download and use.
- The NPM program is installed on your computer when you install Node.js
- It installs the packages you want to use and provides a useful interface to work with them.

Fig 5: Node package Manager

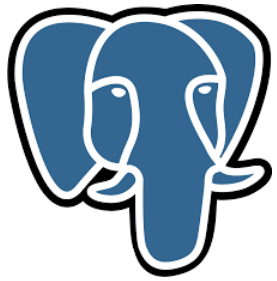
.EJS Files:

- EJS (Embedded JavaScript) is essentially HTML with JavaScript-based features embedded for templating.
- EJS is a JavaScript templating library commonly used for building html strings from JSON data.

Fig 6: .EJS Files

Bootstrap:

- Bootstrap is a free front-end framework for faster and easier web development
- Bootstrap includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many other, as well as optional JavaScript plugins
- Bootstrap also gives you the ability to easily create responsive designs.
- Responsive web design is about creating web sites which automatically adjust themselves to look good on all devices, from small phones to large desktops.



PostgreSQL

- PostgreSQL is a powerful, open source object-oriented relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads

Similar works

- HyPer – A Hybrid OLTP&OLAP High Performance DBMS
- HyPer is a main-memory-based relational DBMS for mixed workloads.
- It is a so-called all-in-one New-SQL database system that entirely deviates from classical disk-based DBMS architectures by introducing many innovative ideas including machine code generation for data-centric query processing and multi-version concurrency control, leading to exceptional performance.

A Hybrid OLTP&OLAP High Performance DBMS

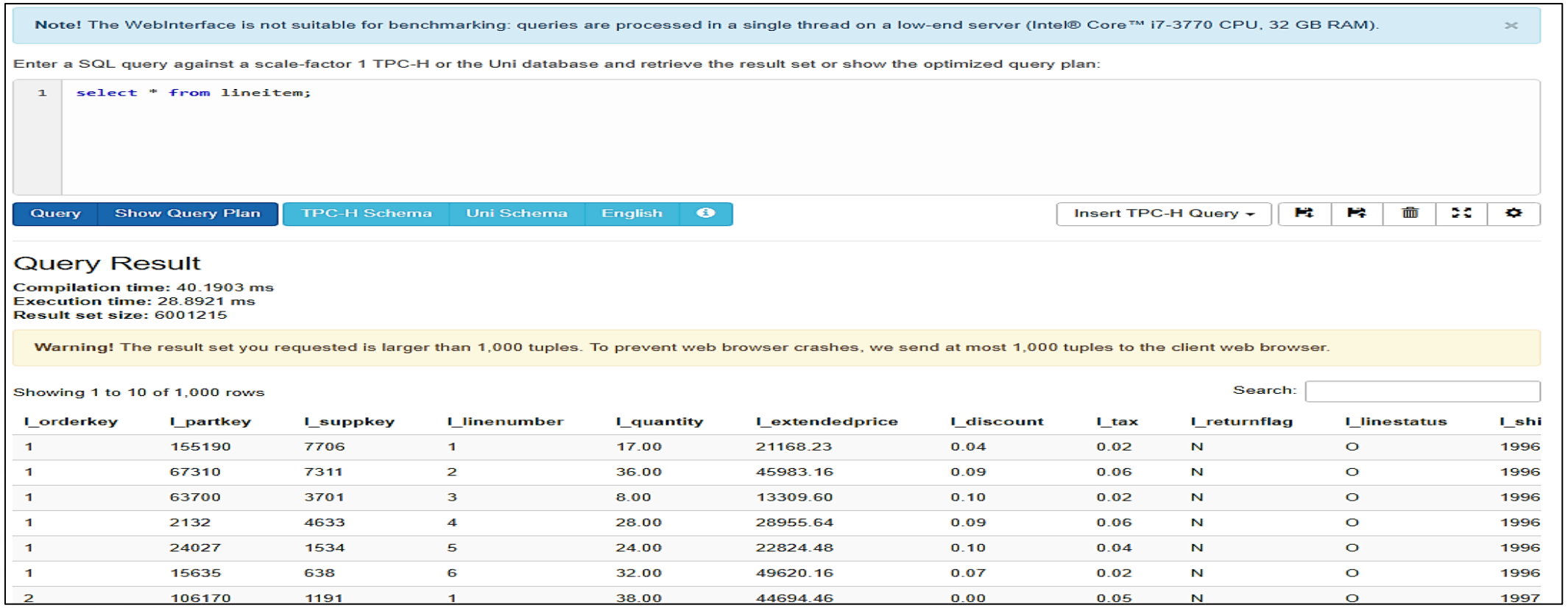


Fig 7 : Hyper web interface

Project idea

- The main idea of the project is to display the results of Correlated Sample Synopsis (CS2) produced sample dataset(s_tpch1g) by taking less planning and execution time compared to the original dataset(tpch1g).
- The front-end web interface shows the text editor to submit a SQL query.
- Once submitted the application displays performance metrics and results of both Postgre and sample Postgre Databases.
- The results are viewed directly on the web page.

Project Flow

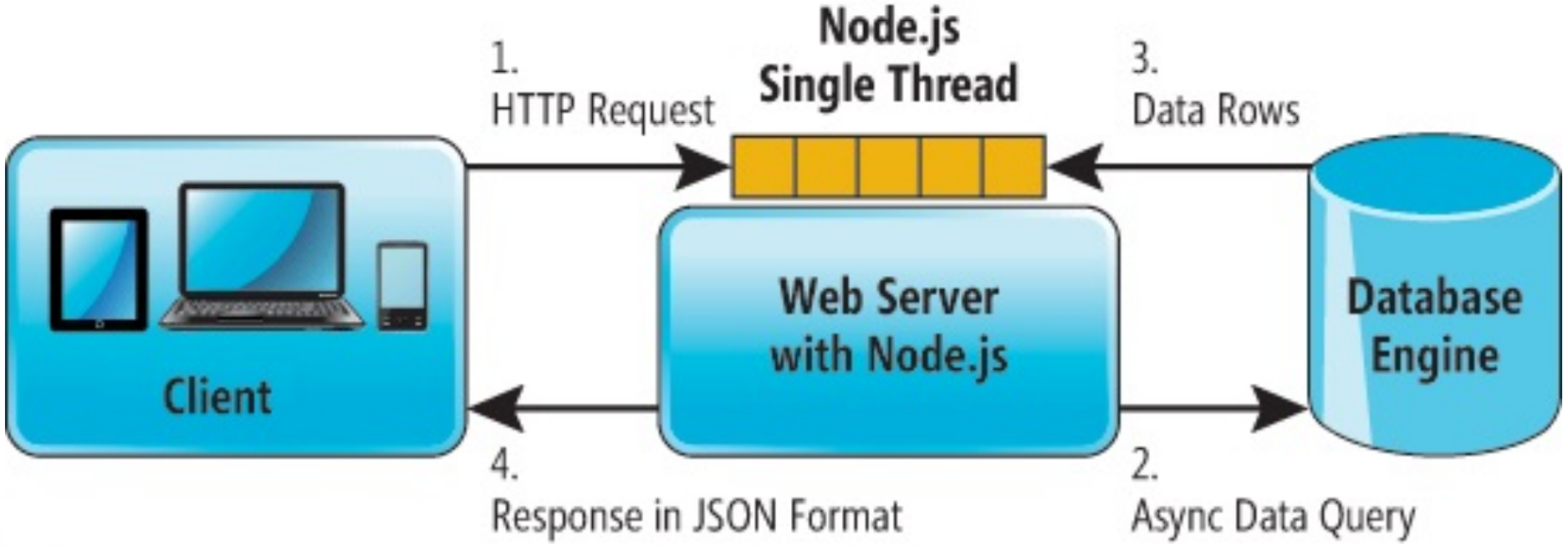


Fig 8 : General process flow

Project Flow Diagram

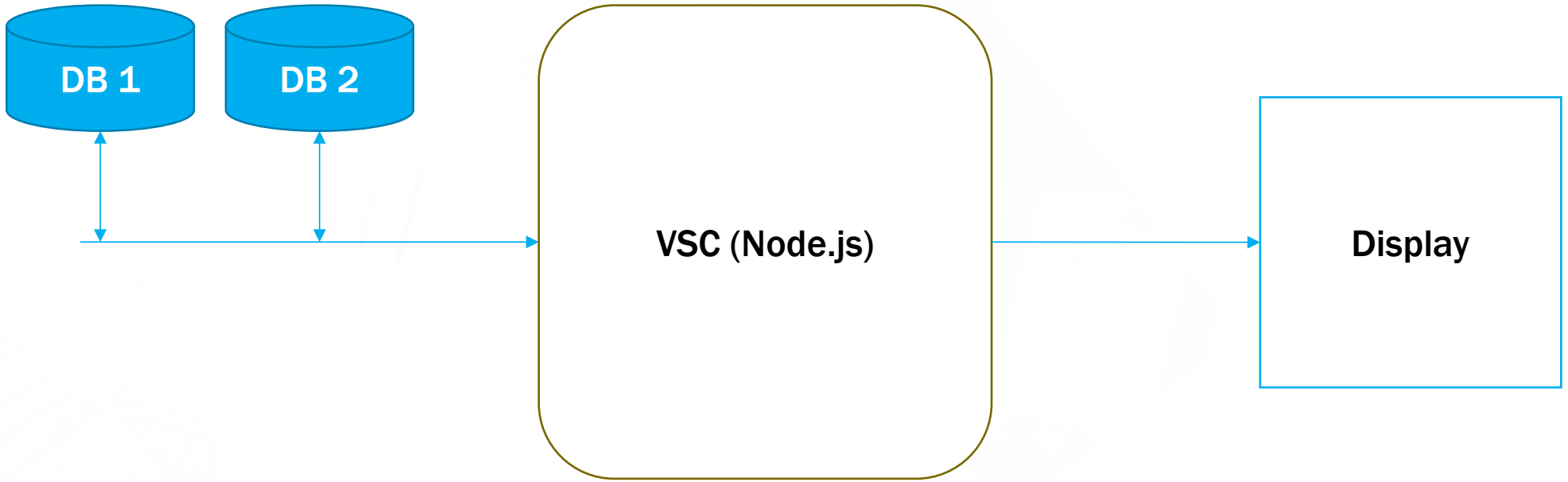


Fig 9 : Actual project flow

Selection of Data Sets

- Transaction Protocol Council Benchmark H (TPC-H)
- Correlated Sample Synopsis (CS2)

Transaction Protocol Council Benchmark H (TPC-H)

- The TPC Benchmark™H (TPC-H) is a decision support benchmark.
- It consists of a suite of business-oriented ad-hoc queries and concurrent data modifications.
- The queries and the data populating the database have been chosen to have broad industry-wide relevance.
- This benchmark illustrates decision support systems that examine large volumes of data, execute queries with a high degree of complexity, and give answers to critical business questions.

- 1 GB TPC-H dataset used (tpch1g)

```
postgres=# \d+
No relations found.
postgres=# \c tpch1g
You are now connected to database "tpch1g" as user "postgres".
tpch1g=# \d+
                    List of relations
 Schema |   Name   | Type  |  Owner   |    Size    | Description
--------+----------+-------+----------+------------+-------------
 public | customer | table | postgres | 28 MB      |
 public | lineitem | table | postgres | 903 MB     |
 public | nation   | table | postgres | 8192 bytes |
 public | orders   | table | postgres | 209 MB     |
 public | part     | table | postgres | 32 MB      |
 public | partsupp | table | postgres | 136 MB     |
 public | region   | table | postgres | 8192 bytes |
(7 rows)
```

Fig 10: Transaction Protocol Council Benchmark H (TPC-H)

Correlated Sample Synopsis (CS2)

- Correlated Sampling aims to gather a sample of joinable sample tuples from joinable relations in the database to form a synopsis.
- Consists of correlated sample tuples that retain join relationships with less storage.
- A statistical technique, called reverse sample, and design a powerful estimator, called reverse estimator, was used to fully utilize correlated sample tuples for query estimation.
- s_tpch1g dataset produced from the 1 GB TPC-H dataset using CS2

```
postgres=# \c s_tpch1g
You are now connected to database "s_tpch1g" as user "postgres".
s_tpch1g=# \d+
                                List of relations
 Schema |   Name   | Type  |  Owner   |    Size    | Description
--------+----------+-------+----------+------------+-------------
 public | customer | table | postgres | 8584 kB    |
 public | lineitem | table | postgres | 9336 kB    |
 public | nation   | table | postgres | 8192 bytes |
 public | orders   | table | postgres | 8504 kB    |
 public | part     | table | postgres | 8536 kB    |
 public | partsupp | table | postgres | 10176 kB   |
 public | region   | table | postgres | 8192 bytes |
 public | supplier | table | postgres | 1808 kB    |
(8 rows)
```

Fig 11: Correlated Sample Synopsis (CS2)

Node js coding in Visual studio code (VSC)

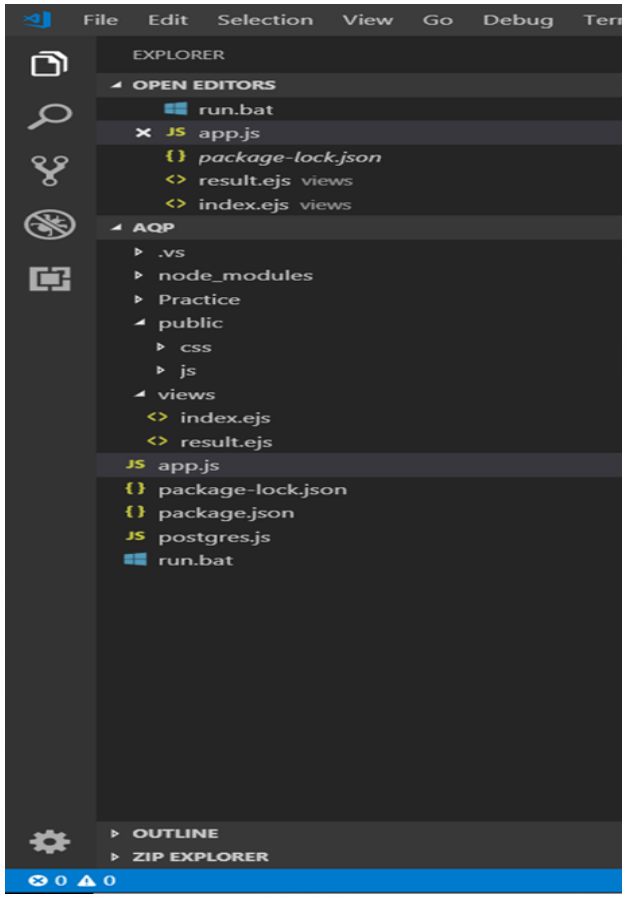


Fig 12: Modules

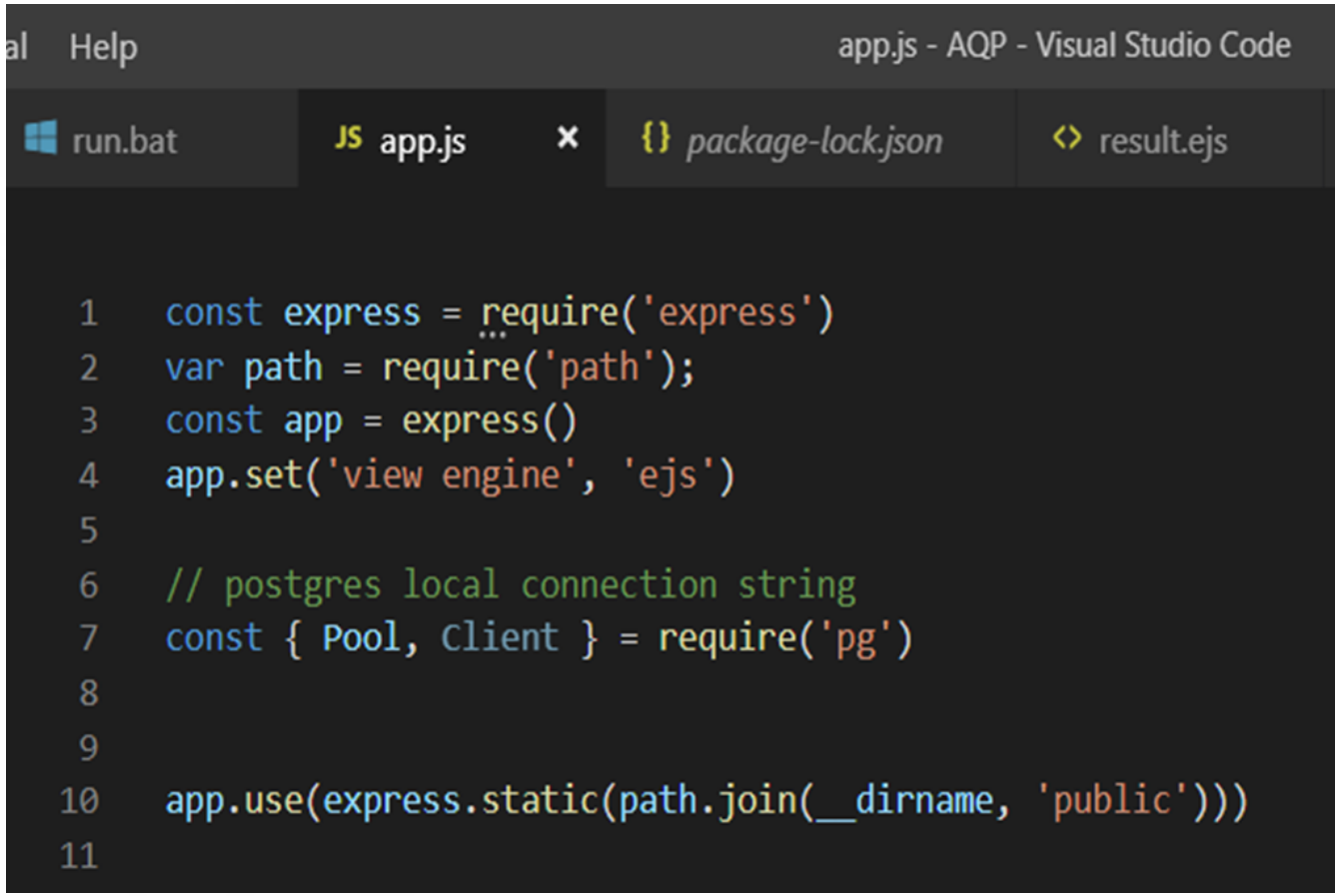


Fig 13: Required packages for app.js

Connection string to postgre databases in VSC

```
pool = new Pool({
    user: 'postgres',
    host: '10.70.0.141',
    database: 'tpch1g',
    password: [redacted]
    port: 5432,
})

GetPostgreStats(query);
GetPostgresResult(query);

samplePool = new Pool({
    user: 'postgres',
    host: '10.70.0.141',
    database: 's_tpch1g',
    password: [redacted]
    port: 5432,
})

GetSamplePostgreStats(query);
GetSamplePostgresResult(query);
```

Fig 14: Connection strings and functions

Display

```
function GetPostgresResult(query)
{
        pool.query(query, (err, res) => {
            if (!err) {
                rowsCount = res.rows.length;
                tableData= res.rows;
                pool.end()
                resultCount ++;
                console.log(resultCount)
            }
            else{
                queryerror = true;
                resultCount ++;
            }
        })

}
```

Fig 15: Function to get result

```
function GetPostgreStats(query)
{
        pool.query('EXPLAIN ANALYZE ' + query , (err, res) => {
            if (!err)
            {
                planningTime = '--';
                executionTime = '--';
                res.rows.forEach(element => {

                    if(element["QUERY PLAN"].includes('Aggregate'))
                        planningTime = element["QUERY PLAN"];

                    if(element["QUERY PLAN"].includes('Total runtime'))
                        executionTime = element["QUERY PLAN"];
                    });
                    resultCount ++;
                    console.log(resultCount)
                }
```

Fig 16: Function to get stats

Display

- Index .ejs: Displays initialize page of webserver without result
- Result.ejs: Displays web page with both performance metrics and results of postgre databases.

```
<html>
<head>
    <title></title>
    <script src="/js/modernizr-2.8.3.js"></script>
    <script src="/js/jquery-3.3.1.js"></script>
    <script src="/js/bootstrap.js"></script>
    <script src="/js/codemirror.js"></script>
    <script src="/js/placeholder.js"></script>
    <script src="https://cdn.datatables.net/1.10.19/js/jquery.dataTables.min.js"></script>
    <link href="/css/codemirror.css" rel="stylesheet" />
    <link href="/css/bootstrap.css" rel="stylesheet"/>
    <link href="/css/site.css" rel="stylesheet"/>
    <link href="https://cdn.datatables.net/1.10.19/css/jquery.dataTables.min.css" rel="stylesheet"/>
</head>
```

Fig 17: Bootstrap plugins

Results and Analysis

```
app.listen(3000, function () {
  console.log('Example app listening on port 3000!')
})
```

Fig 18: Http connection port

```
PS C:\Users\yelam\Desktop\AQP\AQP (5)\AQP> node .\app.js
Example app listening on port 3000!
express deprecated req.param(name): Use req.params, req.body, or req.query instead app.js:22:21
PS C:\Users\yelam\Desktop\AQP\AQP (5)\AQP> node .\app.js
Example app listening on port 3000!
express deprecated req.param(name): Use req.params, req.body, or req.query instead app.js:22:21
```

Fig 19: Node terminal output

Results and Analysis

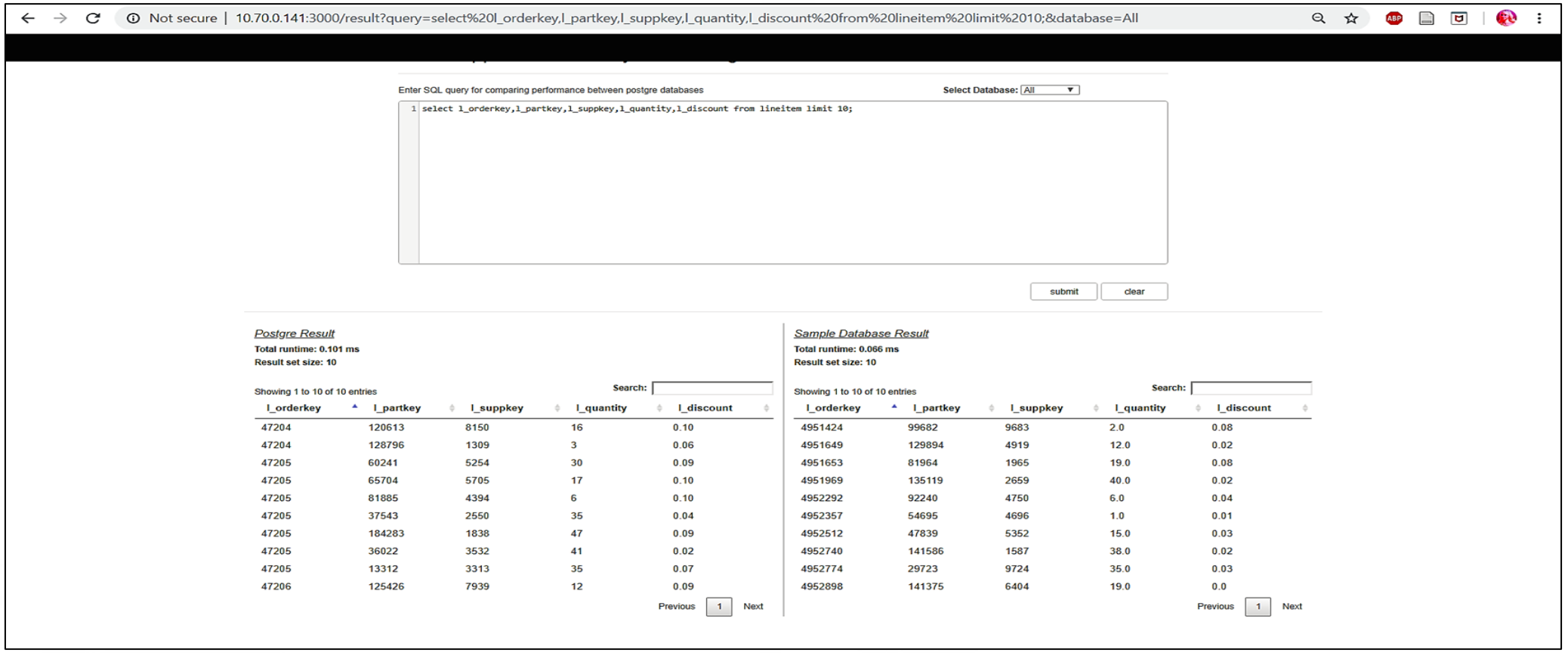


Fig 20: Web page display

Conclusion

- Multiple tools and plugins were used in coding for the web server page.
- TPC-H and CS2 Postgre SQL databases were used in connection string of the application.
- The application is deployed into postgre server
- The web server page obtained through the project display the front-end results of both sample dataset(s_tpch1g) and original tpch1g database.
- The sample dataset(s_tpch1g) takes less planning and execution time compared to the original dataset(tpch1g) by submitting the same SQL query in the text editor.

Acknowledgements

Bharat Yelamanchi

David Wilson

Tasha Wells

Dr. Feng Yu