P-3615

P-3615

PERFORMANCE TESTING OF WEB APPLICATIONS

A PROJECT REPORT

Submitted by

DINESH.V (0710108012)

GOWTHAM.S (0710108015)

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

(An Autonomous Institution Affiliated to Anna University,Coimbatore)

ANNA UNIVERSITY OF TECHNOLOGY, COIMBATORE - 641049

APRIL 2011

BONAFIDE CERTIFICATE

This is to certify that the project report entitled **"PERFORMANCE TESTING OF WEB APPLICATIONS"** is the bonafide work of **"DINESH.V, GOWTHAM.S"** who carried out the research under my supervision.

SIGNATURE

[illegible]

Mr.T.SUDHAKAR,M.E.,

SUPERVISOR

Assistant Professor

Department of Computer Science and Engineering

Kumaraguru College of Technology

Coimbatore-641049

SIGNATURE

[illegible]

Mrs.P.DEVAKI,M.E.,(Ph.D)

HEAD OF THE DEPARTMENT

Department of Computer Science and Engineering

Kumaraguru College of Technology

Coimbatore-641049

[illegible]

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION

We hereby declare that the project entitled " **PERFORMANCE TESTING OF WEB APPLICATIONS**", is a record of original work done by us and to the best of our knowledge, a similar work has not been submitted to Anna University or any Institutions, for fulfillment of the requirement of the course study.

The report is submitted in partial fulfillment of the requirement for the award of the Degree of Bachelor of Computer Science and Engineering of Anna University Of Technology, Coimbatore.

[illegible]

Place: Coimbatore (DINESH.V)

Gowtham.S

Date: 20.04.2011 (GOWTHAM.S)

ACKNOWLEDGEMENT

We extend our sincere thanks to our management, our chairman, **Padmabhusan Arutselvar Dr.N.Mahalingam, B.Sc.,F.I.E.,**for providing us with the necessary facilities and oppurtunities.

We extend our sincere thanks to our Principal,**Dr.S.Ramachandran., Ph.D.,** Kumaraguru College of Technology, Coimbatore, for being a constant source of inspiration and providing us with the necessary facility to work on this project.

We would like to make a special acknowledgement and thanks to **Mrs.P.Devaki,M.E,.(Ph.D).,**Head of Department of Computer Science &Engineering, for her support and encouragement throughout the project.

We express our deep gratitude and gratefulness to **our Guide Mr.T.Sudhakar,M.E.,** Department of Computer Science & Engineering, for his supervision, enduring patience, active involvement and guidance.

We would like to convey our honest thanks to our **class advisor, Mrs. Ramathilagam,M.E.,** and **all the other Faculty members** of the Department for their enthusiasm and wealth of experience from which we have greatly benefited.

We would also like to thank all our **non teaching staff and the lab technicians** for their constant support.

We also thank our **friends and family** who helped us to complete this project fruitfully.

TABLE OF CONTENTS

ABSTRACT

Business growth is being increasingly tied with reliable IT infrastructure. With the heavy business dependence on web based software like SaaS, MTS, ERP, CRM, etc., the need for a reliable, cost-effective and easy to use solution has become more important than ever.

Performance measure is testing that is performed, to determine how fast some aspect of a system performs under a particular workload. It can also serve to validate and verify other quality attributes of the system, such as scalability, reliability and resource usage.

Performance testing is a subset of Performance engineering, an emerging computer science practice which strives to build performance into the design and architecture of a system, prior to the onset of actual coding effort.

The objective of this project is to provide performance/through put of the DB queries and HTTP request. It supports deploy the zip applications can be installed in a silent mode.

Key words: PM – Performance Meter, ZIP - Compressed File, IT – Information Technology, DB – Database, HTTP - Hypertext Transfer Protocol, SaaS – Software as Service, CRM – Customer Relation Management.

LIST OF ABBREVIATIONS USED

SaaS	- Software as Service
API	- Application Programming interface
DB	- Database
HTTP	- Hypertext Transfer Protocol
DAO	- Data Access Object
JDBC	- JAVA Database Connectivity
SDLC	- Software Development Life Cycle
OS	- Operating System
RAM	- Random Access Memory
UI	- User Interface
IP	- Internet Protocol
GUI	- Graphical User Interface

LIST OF FIGURES

LIST OF TABLES

CHAPTER - 1

INTRODUCTION

Performance testing in web application development is a part of **SDLC** (Software Development Life Cycle) which ensures the quality of the product and makes the product highly usable by the customers. The web based performance meter allows to us do testing from the remote atmosphere. We can do the testing/performance measure configuration from remote. With this we can configure same test case in many machines easily, by configuring in one machine and export the configuration and import it to the other test environment from one location.

The objective of this project is to provide performance/through put of the DB queries and HTTP request with the following features.

- Input the queries and database details.
- Input the HTTP request and its input parameters.
- Performance report can be calculated.
- Even load test performance can be done.
- Through put can be calculated.
- It support for database – statement and prepared statement and web application request HTTP

1.1 Key features and benefits

- ➢ Supports Windows 2000, XP, 2003, 2008 and Vista, Linux flavors like Ubuntu, CentOS, SUSE, etc., operating systems.
- ➢ Web-based user interface.
- ➢ Target server can be defined from remote location, if the system has access to the site through intranet.
- ➢ Provides the report for the applied configurations.
- ➢ Affordable web based solution
- ➢ No formal training required for working with the software, minimal performance testing knowledge is enough.

- Enhances productivity of Software tester (mainly performance testers) and reduces their workload.
- Reduces time taken to testing and multiple testing can be carried out in the short span of time.
- Increases the quality of the software development and reduce the cost.
- Easy to install and use.

1.2 Background

Web based software usage is being increased day by day on knowledge sharing, answer to the support raised by the client. In current scenario, lot of Internet application is evolving. If any, web application is developed and that should be highly performing. At the end of every module developed the performance to be tested and if the performance didn't meet the criteria, then performance should be tuned.

To measure the performance of HTTP request and Queries used to generate need a tool. Performance test and load test can be done using our tool Performance meter. This can be done before the module is released to the testing team. This is can be included in the Software development life cycle, before and after functional the testing process.

Performance testing Sub-Genres are Load test, Stress test, Endurance test, Spike test, Configuration test, Isolation test.

Performance meter allow doing the Load test, Stress test, Endurance test, spiking test and Isolation test. According to latest survey, many service provider fails in the performance and pays fine range of million dollars for their fault.

1.2.1 Software Performance Testing

Performance Testing covers a broad range of engineering or functional evaluations where a material, product, system, or person is not specified by detailed material or component specifications: rather, emphasis is on the final measurable performance characteristics. Testing can be a qualitative or quantitative procedure.

Performance Meter is a tool for analyzing and measuring the performance of a variety of services, with a focus on web applications. It can be used as a unit test tool for JDBC Database connections, HTTP and DAO.

1.3 Scope

The scope of the performance testing is as follows.

- Load Testing
- Stress Testing
- Endurance Testing
- Spike Testing
- Isolation Testing

All these testing can be carried out with this project "**Performance Meter**".

1.3.1 Load Testing

Load testing is the simplest form of performance testing. A load test is usually conducted to understand the behavior of the application under a specific expected load. This load can be the expected concurrent number of users on the application performing a specific number of transactions within the set duration. This test will give out the response times of all the important business critical transactions. If the database, application server, etc. are also monitored, then this simple test can itself point towards any bottlenecks in the application software.

1.3.2 Stress Testing

Stress testing is normally used to understand the upper limits of capacity within the application landscape. This kind of test is done to determine the application's robustness in terms of extreme load and helps application administrators to determine if the application will perform sufficiently if the current load goes well above the expected maximum.

1.3.3 Endurance Testing

Endurance testing is usually done to determine if the application can sustain the continuous expected load. During endurance tests, memory utilization is monitored to detect potential leaks. Also important, but often overlooked is performance degradation. That is, to ensure that the throughput and/or response times after some long period of sustained activity are as good as or better than at the beginning of the test.

1.3.4 Spike Test

Spike testing, as the name suggests is done by spiking the number of users and understanding the behavior of the application; whether performance will suffer, the application will fail, or it will be able to handle dramatic changes in load.

1.3.5 Isolation Testing

Isolation testing is not unique to performance testing but a term used to describe repeating a test execution that resulted in an application problem. Often used to isolate and confirm the fault domain.

CHAPTER - 2

2.1 System Requirements

- Hardware Requirements
- Software Requirements

2.1.1 Hardware Requirements

Hardware	Recommended
Processor	P4 - 1.5 GHz
RAM	1 GB
Disk Space	2.5 GB*

*For managing up to 200 computers. The required disk space will go up to 10 GB for managing up to 2000 computers.

2.1.2 Software Requirements

2.1.2.1 Supported Platforms

This supports the following Microsoft Windows operating system versions:

Desktops

- Windows 2000 Professional
- Windows XP Professional
- Windows Vista
- Ubuntu
- CentOS

Servers

- JAVA
- Tomcat

2.1.2.2 Supported Browsers

This requires one of the following browsers to be installed in the system for working with this solution's Client.

- Internet Explorer 5.5 and above
- Firefox
- Netscape 7.0 and above

2.1.2.3 Supported OS Languages - English

Preferred screen resolution 1024 x 768 pixels or higher

2.1.3 Functional Requirements

The following are the list of functional requirements identified and below is an extract from the software requirements specification.

- To test the database server, we need to have the databases server deployed or installed in one host machine. First we have to create the Database, schema if required, and tables with required constraint.
- Collect the database server JDBC connector and include in to the lib directory of the tomcat server. This act as bridge between the Database server and Performance meter.
- Store the configuration which is to be tested in the Performance Meter. Like, Project name, Number of Threads, Number of Loops, Database type, Host name, Port, Database name, User name, Password, Statement Type, Queries.
- Select the project and run the test.
- To test the web server application, we need to have the web server and application to be deployed in the web server, to which the performance testing to be carried out.
- Collect the application details which is to be tested.
- Store the configuration which is to be tested in the Performance Meter. Like, Project name, Number of Threads, Number of Loops, Http method, Host name, Port, URL and its parameters.
- Select the project and run the test.

2.1.4 Non Functional Requirements

2.1.4.1 Scalability

The product should have the ability to handle larger volumes of test that to be managed. This is important because people will go for this product only for large scale performance testing. Hence the product should have the capability of handling and managing the configuration data for the medium and large scale testing.

2.1.4.2 Performance

The initial data loading, storing the configuration data in the database and xml, retrieving the data for viewing configuration reports. It is an important factor, as the user cannot wait for a long time for the product to do the above said actions. The product should have the capability to load the data's as quickly as possible. The metrics for performance is yet to arrive.

2.1.4.3 Extendibility

For this release, this product supports Tomcat server. The product should be easily extendable to support other Web server i.e., Websphere CE etc.,.

The current architecture was designed in such a way that the developers need to implement for any webserver. It will not have any OS/Web server dependency, so that it will be easy to use any web server.

CHAPTER - 3

3.1 High level design (Architectural Design)

This project (Performance Meter) currently supports Http, DB and DAO.

The following is the overall architecture of this project.

There are three major components in this architecture Http, Database and DAO (Direct access Object). The architecture diagram is given below:

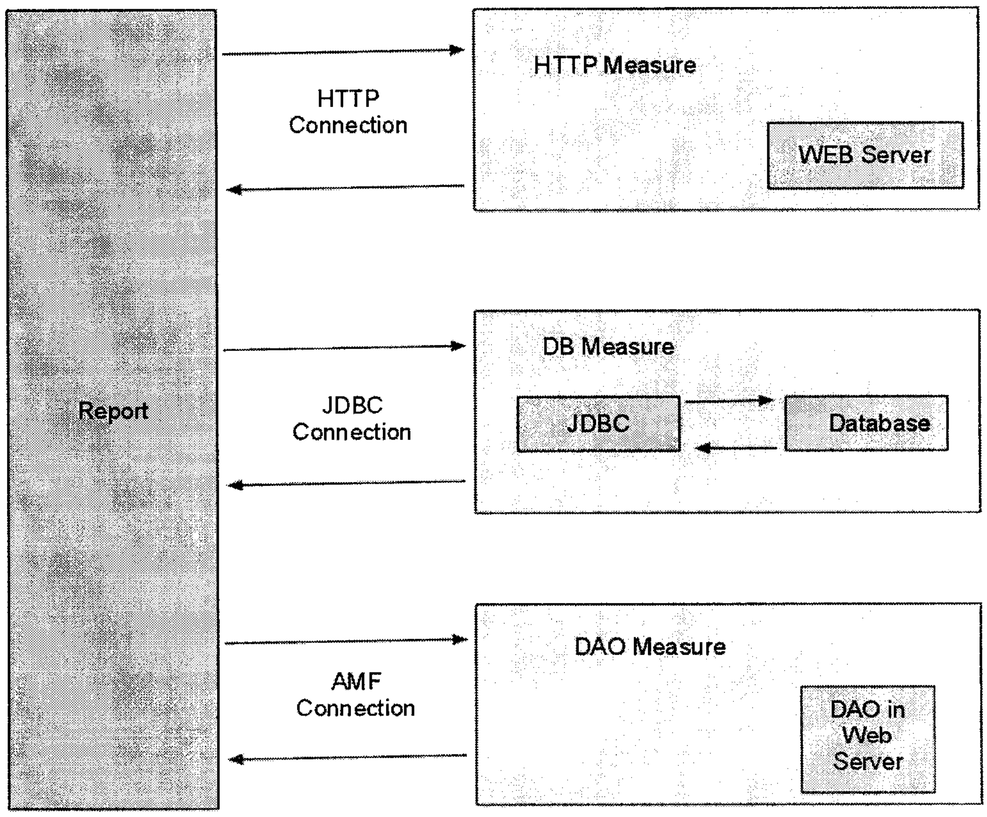


Fig. 3.1 High Level Architecture

3.1.1 Http Module

It is responsible for taking input from the tester to do configuration for testing the HTTP request.

Console: The GUI which will interact with tester, take input for a configuration. Also it will show the project, to run the project or test case. Once the project or test case is executed, the report can be viewed in the Result tab with time take to execute the request and it's through put.

Storage: Store the configuration which is to be tested in the Performance Meter. Like, Project name, Number of Threads, Number of Loops, Http method, Host name, Port, URL and its parameters.

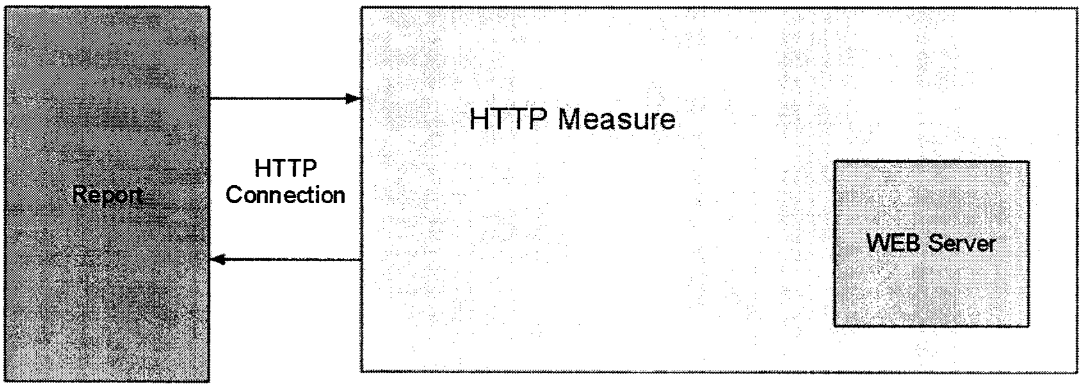


Fig. 3.2 Http Module

3.1.2 DB MODULE

It is responsible for taking input from the tester to do configuration for testing the Database queries.

Console: The GUI which will interact with tester, take input for a configuration. Also it will show the project, to run the project or test case. Once the project or test case is executed, the report can be viewed in the Result tab with time take to execute the request and it's through put.

Storage: Store the configuration which is to be tested in the Performance Meter. Like, Project name, Number of Threads, Number of Loops, Database type, Host name, Port, Database name, User name, Password, Statement Type, Queries.

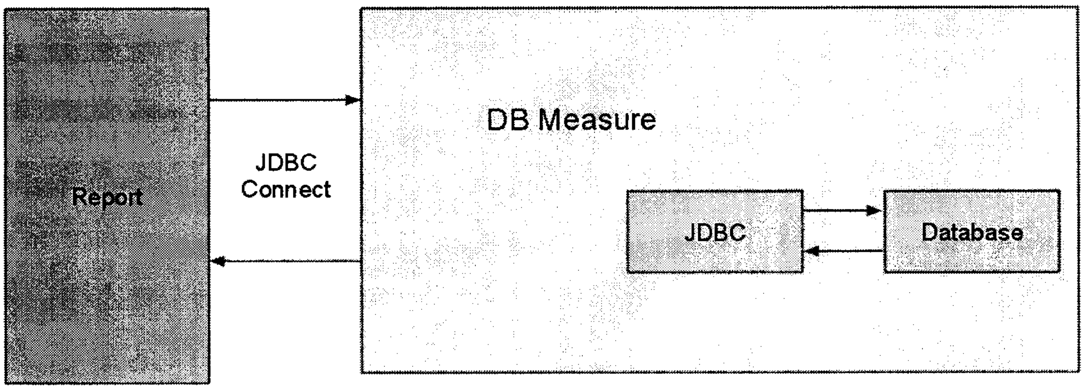


Fig. 3.3 DB Module

3.1.3 DATA ACCESS OBJECT MODULE

It is responsible for taking input from the tester to do configuration for testing the DAO requests.

Console: The GUI which will interact with tester, take input for a configuration. Also it will show the project, to run the project or test case. Once the project or test case is executed, the report can be viewed in the Result tab with time take to execute the request and it's through put.

Storage: Store the configuration which is to be tested in the Performance Meter. Like, Project name, Number of Threads, Number of Loops, Host name, Port, URL and its methods and input parameters

Data Access Object Design Pattern

The Data Access Object (DAO) layer is an essential part of good application architecture. Business applications almost always need access to data from relational or object databases and the Java platform offers many techniques for accessing this data. The oldest and most mature technique is to use the Java Database Connectivity (JDBC) API, which provides the capability to execute SQL queries against a database and then fetch the results, one column at a time

The Benefits of Data Access Objects

The Data Access Object design pattern provides a technique for separating object persistence and data access logic from any particular persistence mechanism or API. There are clear benefits to this approach from an architectural perspective. The Data Access Object approach provides flexibility to change an application's persistence mechanism over time without the need to re-engineer application logic that interacts with the Data Access Object.

Advantages

The advantage of using data access objects is the relatively simple and rigorous separation between two important parts of an application which can and should know almost nothing of each other, and which can be expected to evolve frequently and independently.

The expected benefit of use of Data Access Objects in a Java scenario is:

- Improved efficiency and performance of the data layer since it is standard reusable software.
- Modifications can be made to the DAO implementation without altering other decoupled modules of the application.

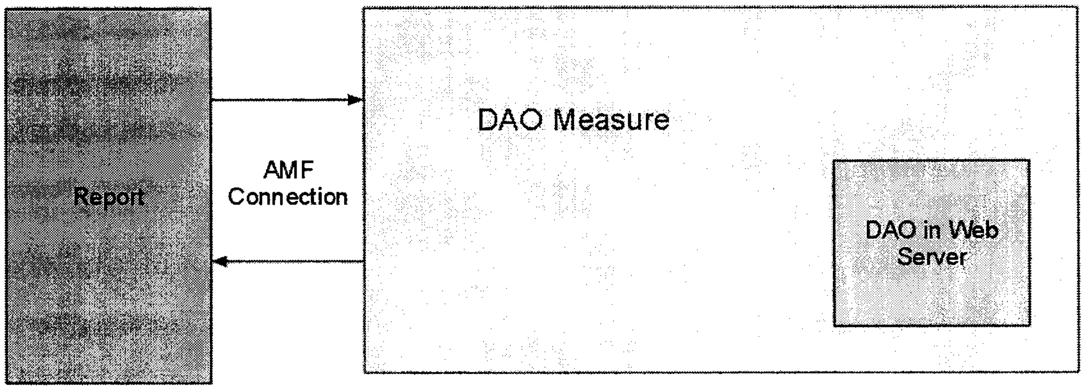


Fig. 3.4 DAO Module

CHAPTER - 4

4.1 Low Level Design

4.1.1 Steps to configure the Performance Meter

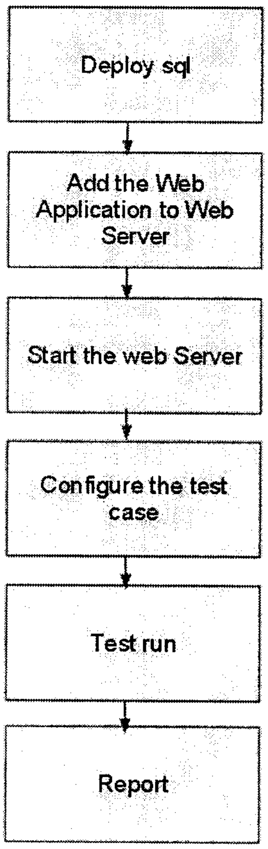


Fig. 4.1 Steps to configure the Performance Meter

Steps to configure the Performance Meter

Deploy SQL

Execute the sql file to the embedded database in the Performance meter. This will create a database and tables to store the data in the database. The data will be persisted in this embedded database like project or test cases, reports etc.,.

Deploy Web application

Copy the performance meter web application folder into the tomcat web server. Now performance meter is ready for the use.

4.2 State Flow Diagram

`This chapter explains the various states of configurations and the possible operations can be performed in each state.

There are 3 states, defined in this system to represent the configurations state. All the configurations defined in the system must possess one of the below states defined.

1. **Database Deployment** – Open the console, execute the database sql in the MYSQL server. Once the sql executed, the database server is ready for the performance meter persistence. Now the performance meter web application can be deployed.

2. **Web application Deployment** – Open the webapps folder in the tomcat server, copy the pmeter folder to this folder. Configure the connection pooling to this web application. Copy the required jar file to the lib directory. Now the performance meter is ready for performance testing to the tester.

3. **Configure and Execute the Test case** – Configure the test cases and execute the test case or project to be executed. To do this web application, we can use the user interface of the performance web application.

4.2.1 State Flow Chart

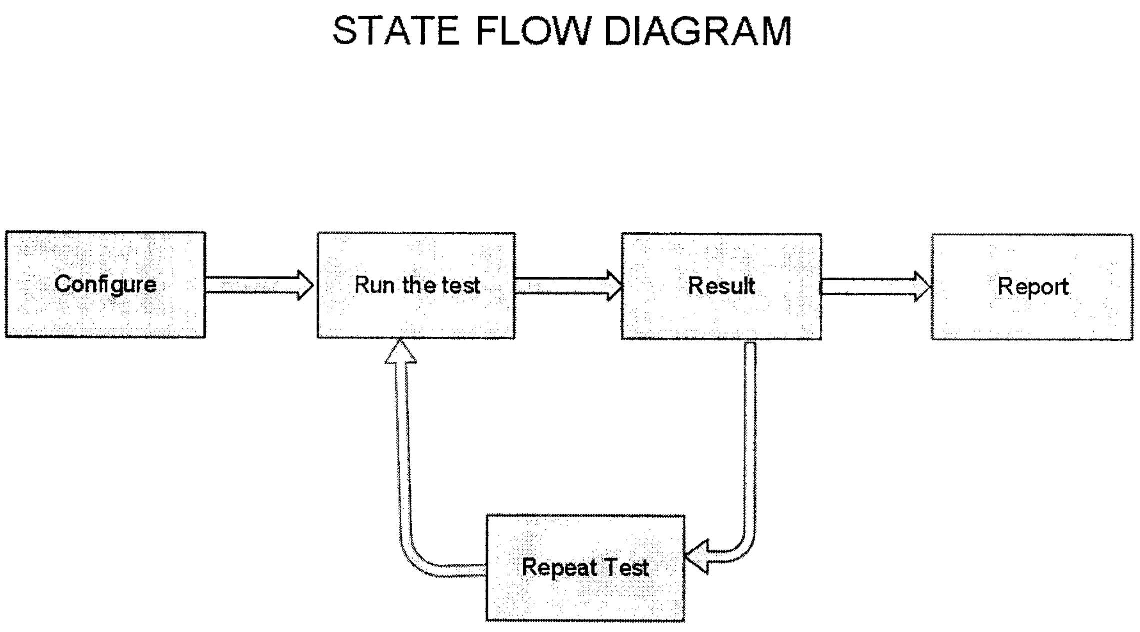


Fig. 4.2 State Flow Diagram

CHAPTER 5

5.1 Database Design

We need to store the data's in the database that are extracted from the configuration test case module. This section deals with database tables that to be created for this product.

In general, the above configurations can be broadly classified into configuration the project and Project reports. This will help to retrieve the stored project and reports generated to corresponding project.

Model the tables separately based in the data structure involved in each of the configuration. This is the approach followed in the overall data model. At the high level, the entire data of Performance Meter can be categorized as specified in the below diagram.

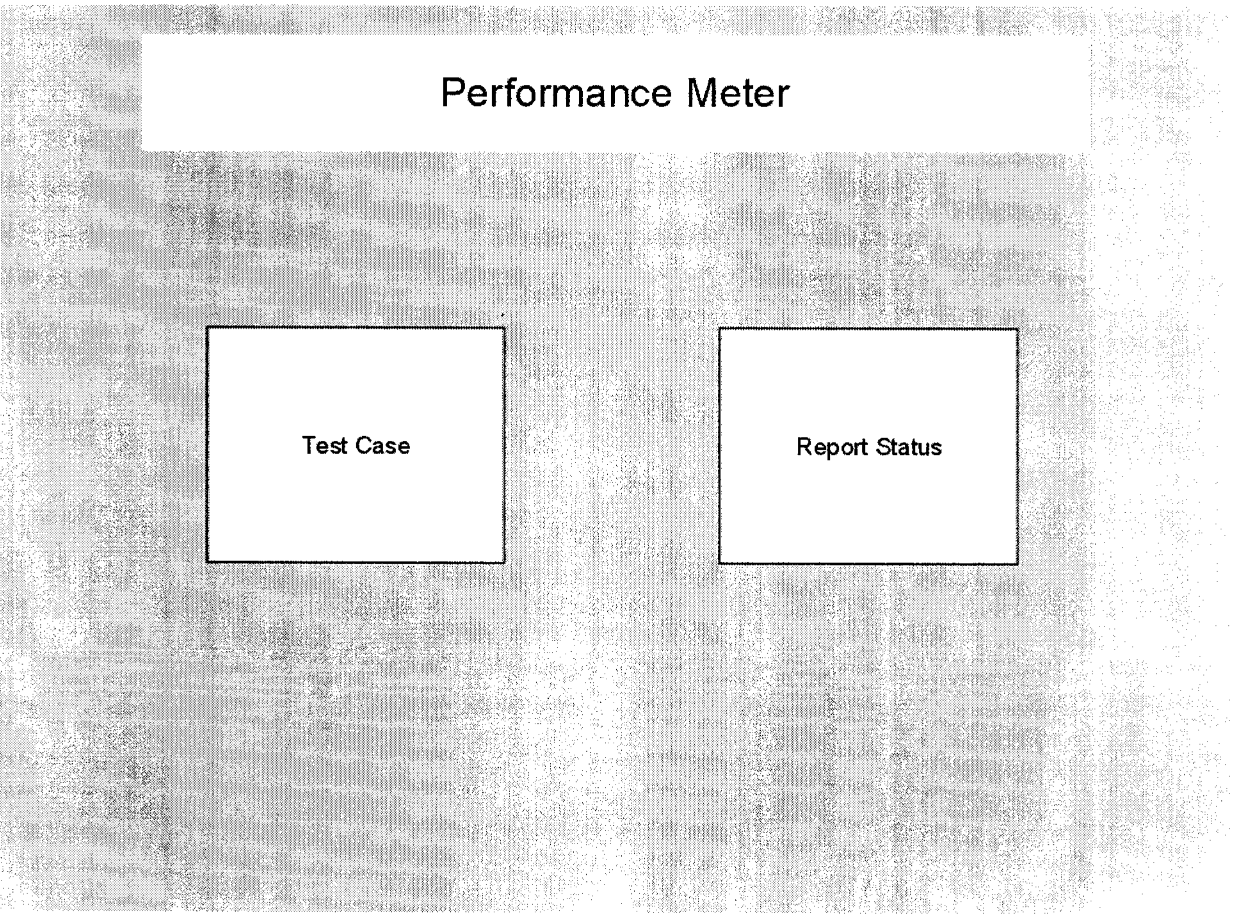


Fig. 5.1 Database Design diagram

A brief explanation of each item is given below:

Test Case

Test Case captures the default data of the project of each configuration. The default settings are nothing but the immediate status.

Report Status

Report Status captures the project reports and to display. This table is referenced in the Test case table, to reference with project.

5.2 Configuration Data – Tables Description

Below is the list of tables used to store the details required for the configuration data.

- Test_Case
- Report_Status

5.2.1 Test_Case

This table contains the project details defined by the user

Column Name	**Description**
PROJECT_ID	Unique Id for each project
PROJECT_NAME	Name of the Project
FILENAME	File name of the test case configuration
STATUS	Status of the project
PROJECT_TYPE	DB/HTTP/DAO

5.2.2 Report_Status

This table contains the Report status which is referenced test case details defined by the user.

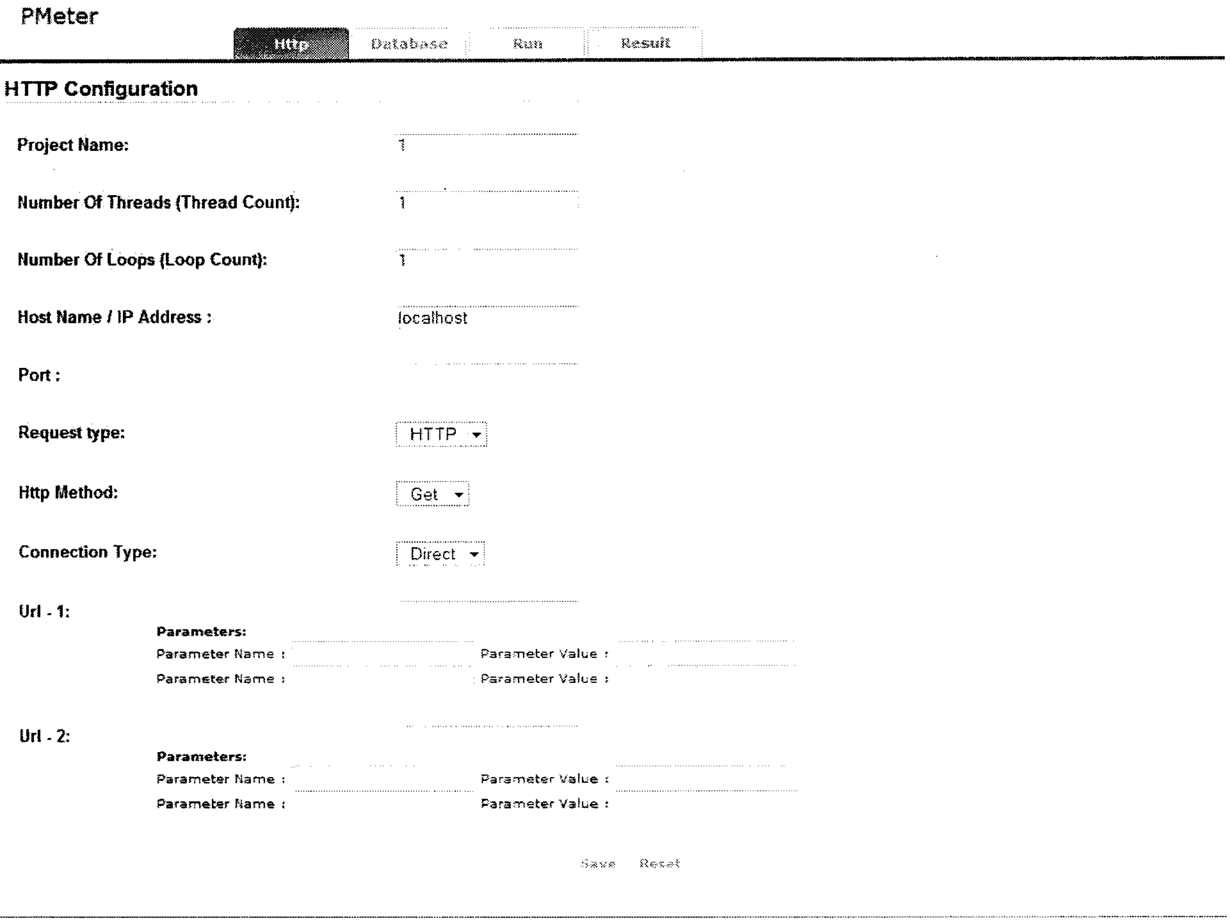
Column Name	Description
REPORT_ID	Unique Id for each report
REPORT_NAME	Name of the Report
STATUS	Report Status
PROJECT_ID	Id from Project
FILENAME	File name of the Report

5.3 User Interface Design

This web-based software to test the performance of the Database queries, Http request, and DAO request. The web application is deployed in the Tomcat server with two type of storage used. The storage servers are database server and file server. The UI have HTTP configuration, DB Configuration, Project list and Result of the project, which test is executed.

5. 3.1 HTTP And DAO Configuration

Http Configuration is to configure the HTTP and DAO test case to be executed. The below figure show the UI for the HTTP Configuration.


PMeter

HTTP Configuration

Project Name: 1

Number Of Threads (Thread Count): 1

Number Of Loops (Loop Count): 1

Host Name / IP Address : localhost

Port :

Request type: HTTP

Http Method: Get

Connection Type: Direct

Url - 1:

Parameters:

Parameter Name : Parameter Value :

Parameter Name : Parameter Value :

Url - 2:

Parameters:

Parameter Name : Parameter Value :

Parameter Name : Parameter Value :

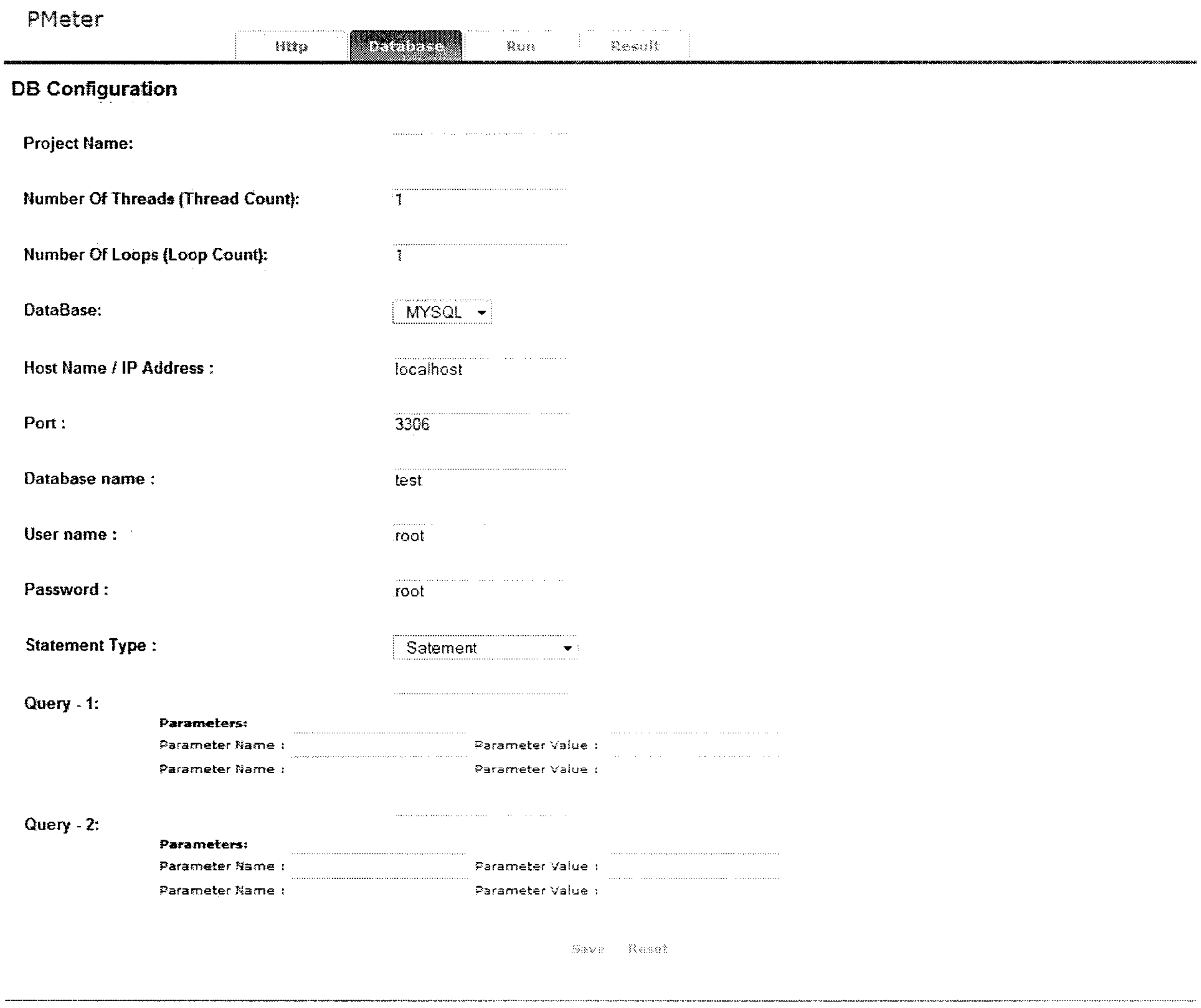
Fig. 5.2 Http Configuration

The below is the step to configure the HTTP/DAO Configuration in Performance Meter.

1. Create a Project name.
2. Number of Threads to be configured.
3. Number of loops to be configured.
4. Configure either host name or IP address of the server to be tested.
5. Configure the port number.
6. Request type either GET or POST or DAO.
7. Configure the URL and its parameter to this.
8. Save this project.

5. 3.2 DB Configuration

DB Configuration is to configure the DB test case to be executed. The below figure show the UI for the DB Configuration.



DB Configuration

Project Name:	
Number Of Threads (Thread Count):	1
Number Of Loops (Loop Count):	1
DataBase:	MYSQL
Host Name / IP Address :	localhost
Port :	3306
Database name :	test
User name :	root
Password :	root
Statement Type :	Satement

Query - 1:

Parameters:

Parameter Name : Parameter Value :

Parameter Name : Parameter Value :

Query - 2:

Parameters:

Parameter Name : Parameter Value :

Parameter Name : Parameter Value :

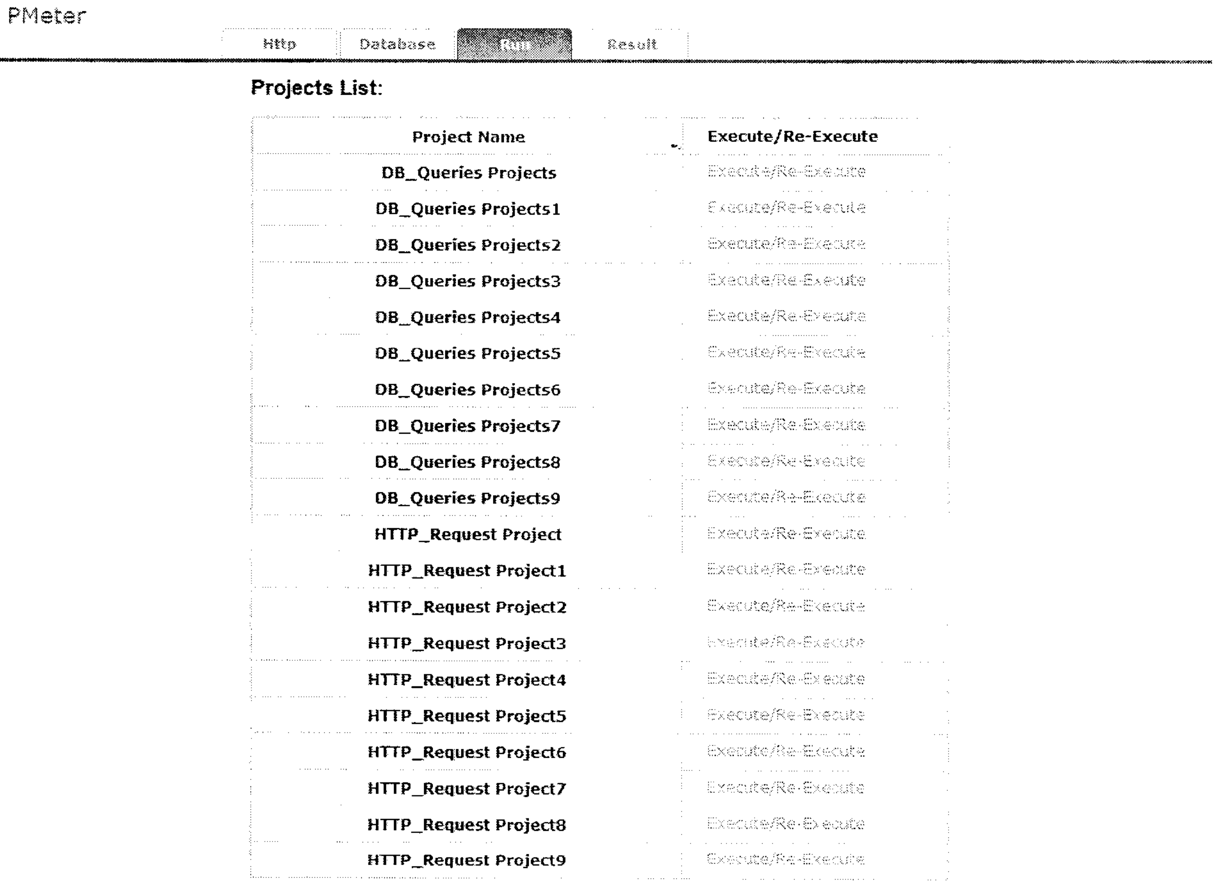
Fig. 5.3 DB Configuration

The below is the step to configure the Http Configuration in Performance Meter.

1. Create a Project name.
2. Number of Threads to be configured.
3. Number of loops to be configured.
4. Select the database.
5. Configure either host name or IP address of the server to be tested.
6. Configure the port number.
7. Configure the Database name.
8. Configure the User name.
9. Configure the Password.
10. Select the Statement Type.
11. Configure the queries and its parameter to this.
12. Save this project.

5. 3.3 Run / Execute the test cases:

The projects will be listed in this tab. The option will be given to execute or re-execute the test cases.



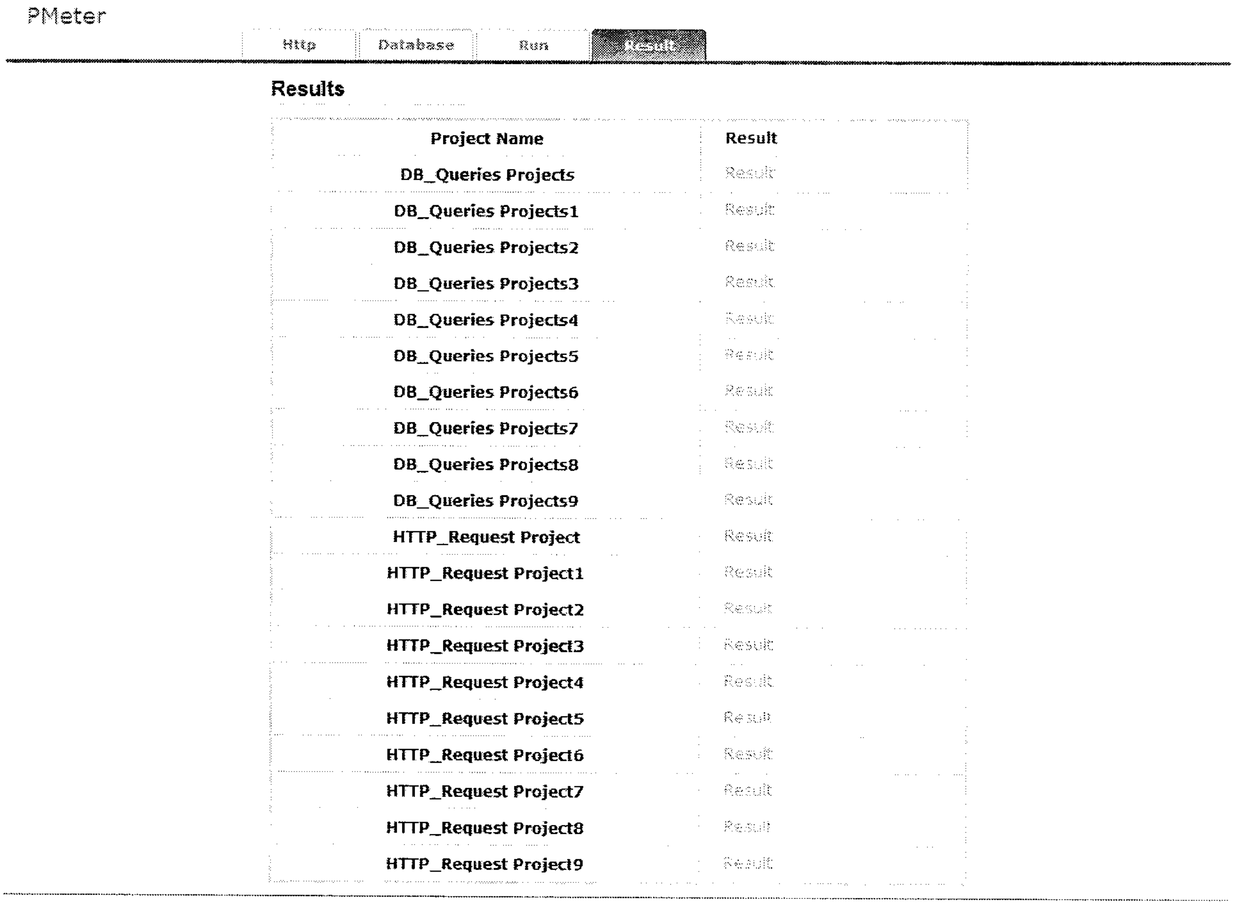
Projects List:

Project Name	Execute/Re-Execute
DB_Queries Projects	Execute/Re-Execute
DB_Queries Projects1	Execute/Re-Execute
DB_Queries Projects2	Execute/Re-Execute
DB_Queries Projects3	Execute/Re-Execute
DB_Queries Projects4	Execute/Re-Execute
DB_Queries Projects5	Execute/Re-Execute
DB_Queries Projects6	Execute/Re-Execute
DB_Queries Projects7	Execute/Re-Execute
DB_Queries Projects8	Execute/Re-Execute
DB_Queries Projects9	Execute/Re-Execute
HTTP_Request Project	Execute/Re-Execute
HTTP_Request Project1	Execute/Re-Execute
HTTP_Request Project2	Execute/Re-Execute
HTTP_Request Project3	Execute/Re-Execute
HTTP_Request Project4	Execute/Re-Execute
HTTP_Request Project5	Execute/Re-Execute
HTTP_Request Project6	Execute/Re-Execute
HTTP_Request Project7	Execute/Re-Execute
HTTP_Request Project8	Execute/Re-Execute
HTTP_Request Project9	Execute/Re-Execute

Fig. 5.4 Project/Test Case List for Execute

5. 3.4 Result:

The projects will be listed in this tab. The option will be given to view the result. Link to view the result will be given.

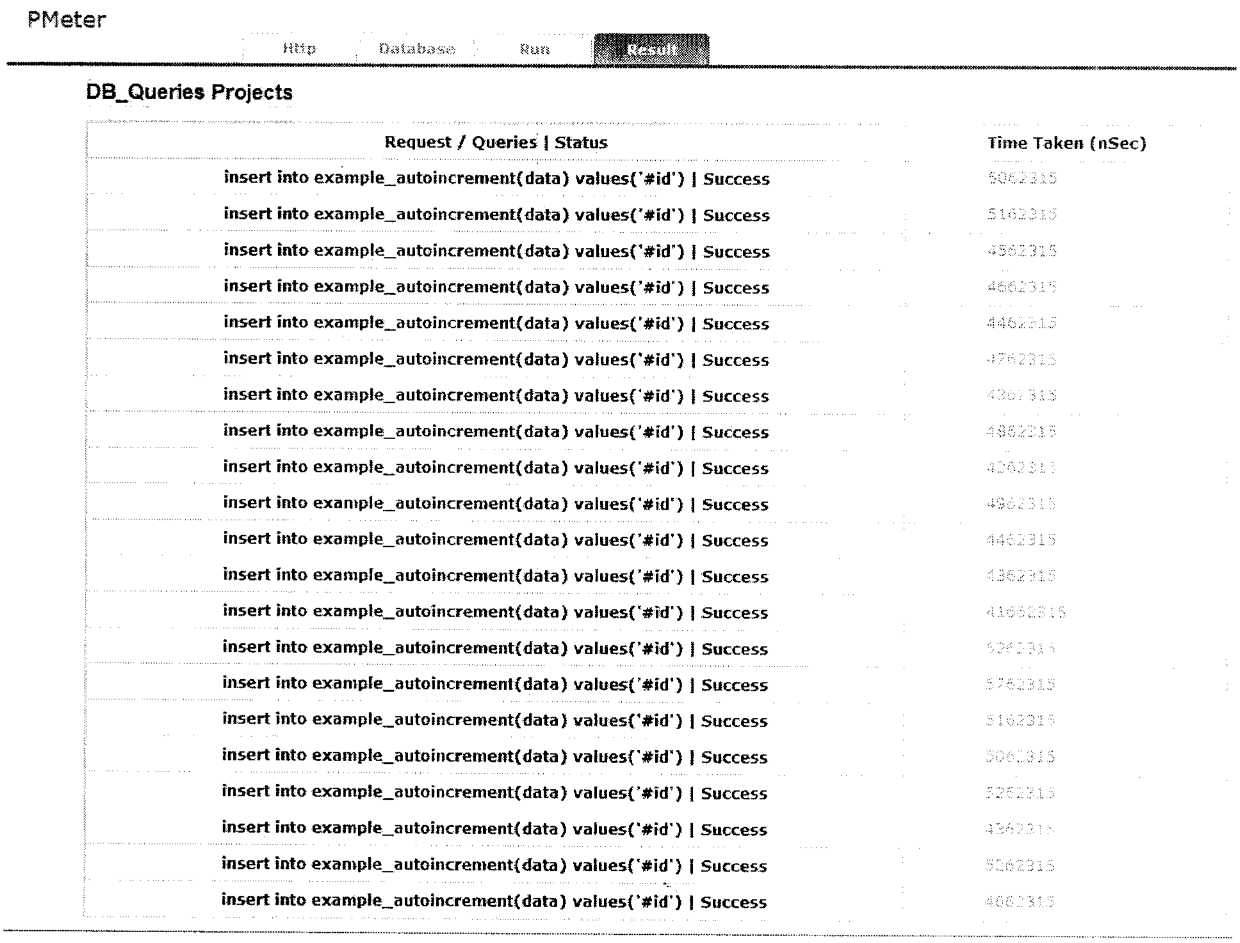


Results

Project Name	Result
DB_Queries Projects	Result
DB_Queries Projects1	Result
DB_Queries Projects2	Result
DB_Queries Projects3	Result
DB_Queries Projects4	Result
DB_Queries Projects5	Result
DB_Queries Projects6	Result
DB_Queries Projects7	Result
DB_Queries Projects8	Result
DB_Queries Projects9	Result
HTTP_Request Project	Result
HTTP_Request Project1	Result
HTTP_Request Project2	Result
HTTP_Request Project3	Result
HTTP_Request Project4	Result
HTTP_Request Project5	Result
HTTP_Request Project6	Result
HTTP_Request Project7	Result
HTTP_Request Project8	Result
HTTP_Request Project9	Result

Fig. 5.5 Project/Test Case for Result

Result for the first test case/project "**DB_Queries Projects**" is as below.



DB_Queries Projects

Request / Queries | Status	Time Taken (nSec)
insert into example_autoincrement(data) values('#id') | Success	5062315
insert into example_autoincrement(data) values('#id') | Success	5162315
insert into example_autoincrement(data) values('#id') | Success	4562315
insert into example_autoincrement(data) values('#id') | Success	4662315
insert into example_autoincrement(data) values('#id') | Success	4462315
insert into example_autoincrement(data) values('#id') | Success	4762315
insert into example_autoincrement(data) values('#id') | Success	4362315
insert into example_autoincrement(data) values('#id') | Success	4862315
insert into example_autoincrement(data) values('#id') | Success	4262315
insert into example_autoincrement(data) values('#id') | Success	4962315
insert into example_autoincrement(data) values('#id') | Success	4462315
insert into example_autoincrement(data) values('#id') | Success	4362315
insert into example_autoincrement(data) values('#id') | Success	41662315
insert into example_autoincrement(data) values('#id') | Success	5262315
insert into example_autoincrement(data) values('#id') | Success	5762315
insert into example_autoincrement(data) values('#id') | Success	5162315
insert into example_autoincrement(data) values('#id') | Success	5062315
insert into example_autoincrement(data) values('#id') | Success	5262315
insert into example_autoincrement(data) values('#id') | Success	4562315
insert into example_autoincrement(data) values('#id') | Success	5062315
insert into example_autoincrement(data) values('#id') | Success	4662315

Fig. 5.6 View Result for the Project

CHAPTER - 6

6.1 CODING

EXECUTE HTTP.JAVA:

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package com.pmeter.executor;

import com.pmeter.xmlparser.RequestUrl;
import com.pmeter.xmlparser.UrlParameters;
import java.util.Collection;
import com.pmeter.db.DBConnection;
import com.pmeter.xmlparser.DataHandler;
import com.pmeter.xmlparser.HttpObject;
import com.pmeter.xmlparser.ResultBundle;
import java.sql.Connection;

import java.io.*;
import java.net.*;

/**
 *
 * @author sony
 */

public class ExecuteHttp extends Thread
{


   private Collection<RequestUrl> urls = null;
   private String method = "get";
   private String hostname = null;
   private String portnumber = null;
   private DataHandler statDataHandler = null;

   public void run()
   {
      if("get".equalsIgnoreCase(this.method))
      {
         executeGet();
      }
      if("post".equalsIgnoreCase(this.method))
```

```
    {
       executePost();
    }

}

public ExecuteHttp(DataHandler dataHandler, HttpObject httpObject, String pmethod)
{

    this.method = pmethod;
    this.urls = httpObject.getUrlsObject();
    this.hostname = httpObject.getHostName();
    this.portnumber = httpObject.getPort();
    this.statDataHandler = dataHandler;
}

private void executeGet()
{
    String urlString = null;
    int code = -1;
    long executionTime = 0;
    for(RequestUrl httpUrl : urls)
    {
       ResultBundle resultBundle = new ResultBundle();
       try
       {
          urlString = "http://"+this.hostname+this.portnumber+httpUrl.getURL().toLowerCase();

          boolean isFirstParaAppend = false;
          for(UrlParameters urlParameter : httpUrl.getParameters())
          {
             String variableName = urlParameter.getUrlParameterName();
             String variableValue = urlParameter.getUrlParameterValue();
             if(!isFirstParaAppend)
             {
                urlString += "?" +variableName+"="+variableValue;
                isFirstParaAppend = !isFirstParaAppend;
             }
             else
             {
                urlString += "&" +variableName+"="+variableValue;
             }
          }
          long startTime = System.nanoTime();

          urlString = urlString.replace ( " ","%20" ) ;
          URL u = new URL ( urlString ) ;
          HttpURLConnection huc =  ( HttpURLConnection )  u.openConnection (  ) ;
          huc.setRequestMethod ( "GET" ) ;
          huc.connect (  ) ;
          urlString = urlString.replaceAll ( "%20","" ) ;
          InputStream is = huc.getInputStream (  ) ;
          code = huc.getResponseCode () ;
          if  ( code == HttpURLConnection.HTTP_OK )
          {
             byte [  ]  buffer = new byte [ 4096 ] ;
             int totBytes, bytes, sumBytes = 0;
             totBytes = huc.getContentLength (  ) ;
             while  ( true )
             {
               bytes = is.read ( buffer ) ;
               if( bytes <= 0 ) break;
               sumBytes+= bytes;
             }
          }
          is.close();
          huc.disconnect (  ) ;
          long endTime = System.nanoTime();
          executionTime = endTime - startTime;

          System.out.println(urlString+" | "+ code + " | " + executionTime +" nS");
       }
       catch(Exception e)
       {
          System.out.println(urlString+" | "+ code + " | " + executionTime+" nS");
       }
       resultBundle.setStatement(urlString);
       resultBundle.setStatus(new Integer(code).toString());
       resultBundle.setTimeTaken(executionTime);
       statDataHandler.addResultBundle(resultBundle);
    }
    statDataHandler.signOFF();

}

private void executePost()
{
    for(RequestUrl httpUrl : urls)
    {
       String urlString = null;
       int code = -1;
       long executionTime = 0;
       ResultBundle resultBundle = new ResultBundle();
       try
```

```
{

  urlString = "http://"+this.hostname+this.portnumber+httpUrl.getURL().toLowerCase();
  boolean isFirstParaAppend = false;
  String params = "";
  for(UrlParameters urlParameter : httpUrl.getParameters())
  {
     String variableName = urlParameter.getUrlParameterName();
     String variableValue = urlParameter.getUrlParameterValue();
     if(!isFirstParaAppend)
     {
        params += variableName+"="+variableValue;
        isFirstParaAppend = !isFirstParaAppend;
     }
     else
     {
        params += "&"+variableName+"="+variableValue;
     }
  }
  long startTime = System.nanoTime();
  System.out.println("params = " + params);
  urlString = urlString.replace ( " ","%20" ) ;
  URL u = new URL ( urlString ) ;
  HttpURLConnection huc =  ( HttpURLConnection )  u.openConnection (  ) ;
  huc.setRequestMethod ( "POST" ) ;
  huc.setDoOutput(true);
  huc.setRequestProperty("User-Agent", "Profile/MIDP-2.0 Configuration/CLDC-1.0");
  huc.setRequestProperty("Content-Language", "en-US");
  huc.connect();
  urlString = urlString.replaceAll ( "%20","" ) ;
  OutputStream os = huc.getOutputStream();
  os.write(params.getBytes());
  InputStream is = huc.getInputStream () ;

  code = huc.getResponseCode () ;
  if  ( code == HttpURLConnection.HTTP_OK )
  {
     byte [  ]  buffer = new byte [ 4096 ] ;
     int totBytes, bytes, sumBytes = 0;
     totBytes = huc.getContentLength (  ) ;
     while  ( true )
     {
       bytes = is.read ( buffer ) ;
       if( bytes <= 0 ) break;
       sumBytes+= bytes;
     }
  }
```

```
          os.close();
          is.close();
          huc.disconnect ( ) ;
          long endTime = System.nanoTime();
          executionTime = endTime - startTime;
        }
        catch(Exception e)
        {
          e.printStackTrace();
        }

        resultBundle.setStatement(urlString);
        resultBundle.setStatus(new Integer(code).toString());
        resultBundle.setTimeTaken(executionTime);
        statDataHandler.addResultBundle(resultBundle);
        System.out.println(urlString+" | "+ code + " | " + executionTime+" nS");

      }
      statDataHandler.signOFF();

    }

}
```

EXECUTE STATEMENT:

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package com.pmeter.executor;

import com.pmeter.xmlparser.DataHandler;
import com.pmeter.xmlparser.Queries;
import com.pmeter.xmlparser.QueryParameters;
import com.pmeter.xmlparser.ResultBundle;
import java.sql.*;
import java.util.Collection;

/**
 *
 * @author sony
 */
public class ExecuteStatement extends Thread
{
   private Connection connection = null;
private Collection<Queries> queries = null;
private String statementType = "statement";
private DataHandler statDataHandler = null;
public void run()
{
   if("statement".equalsIgnoreCase(this.statementType))
   {
      executeStatement();
   }
   if("preparedstatement".equalsIgnoreCase(this.statementType))
   {
      executePreparedStatement();
   }

}

public      ExecuteStatement(DataHandler      dataHandler,      Connection      connection,
Collection<Queries> queries, String statementType)
{
   this.statementType = statementType;
   this.connection = connection;
   this.queries = queries;
   this.statDataHandler = dataHandler;

}

private void executeStatement()
{

   try
   {
      Statement stmt = connection.createStatement();

      for(Queries query : queries)
      {
         String sql = query.getQuery().toLowerCase();
         ResultBundle resultBundle = new ResultBundle();
         for(QueryParameters queryParameter : query.getQueryParameters())
         {

            String variableName = queryParameter.getQueryParameterName();
            String variableValue = queryParameter.getQueryParameterValue();
            sql = sql.replaceAll(variableName, variableValue);
         }
         long startTime = System.nanoTime();
         String response = null;
         try
```

```
        {
          int selectIndex = sql.indexOf("select");

          if(selectIndex== 0 || selectIndex==1)
          {
            ResultSet rs = stmt.executeQuery(sql);
          }
          else{
            stmt.executeUpdate(sql);
          }
            response = "Done";
        }
        catch(Exception e)
        {
          e.printStackTrace();
          response = "Error" ;
        }

        long endTime = System.nanoTime();
        long executionTime = endTime - startTime;
        resultBundle.setStatement(sql);
        resultBundle.setStatus(response);
        resultBundle.setTimeTaken(executionTime);
        statDataHandler.addResultBundle(resultBundle);
        System.out.println(sql+" | "+ response + ": " + executionTime);
      }
    }
    catch(Exception e)
    {
      e.printStackTrace();
    }
    statDataHandler.signOFF();

}

private void executePreparedStatement()
{

    try
    {
      PreparedStatement prepStmt = null;

      for(Queries query : queries)
      {
        ResultBundle resultBundle = new ResultBundle();
        String sql = query.getQuery().toLowerCase();
        prepStmt = connection.prepareStatement(sql);
```

```
int index = 1;
for(QueryParameters queryParameter : query.getQueryParameters())
{
   String fieldType = queryParameter.getQueryParameterName();
   String fieldValue = queryParameter.getQueryParameterValue();
   if("String".equalsIgnoreCase(fieldType)){
      prepStmt.setString(index, fieldValue);
   }
   if("Int".equalsIgnoreCase(fieldType)){
      prepStmt.setInt(index, Integer.parseInt(fieldValue));
   }
   if("Long".equalsIgnoreCase(fieldType)){
      prepStmt.setLong(index, Long.parseLong(fieldValue));
   }
   if("Date".equalsIgnoreCase(fieldType)){
      prepStmt.setDate(index, Date.valueOf(fieldValue));
   }
   if("Double".equalsIgnoreCase(fieldType)){
      prepStmt.setDouble(index, Double.parseDouble(fieldValue));
   }
   if("Float".equalsIgnoreCase(fieldType)){
      prepStmt.setFloat(index, Float.parseFloat(fieldValue));
   }

   index++;
}
long startTime = System.nanoTime();
String response = null;
try
{
   int selectIndex = sql.indexOf("select");
   if(selectIndex== 0 || selectIndex==1)
   {
      prepStmt.execute();
   }
   else
   {
      prepStmt.executeUpdate();
   }
      response = "Done";
}
catch(Exception e)
{
   e.printStackTrace();
   response = "Error" ;
}
```

```
            long endTime = System.nanoTime();
            long executionTime = endTime - startTime;
            resultBundle.setStatement(sql);
            resultBundle.setStatus(response);
            resultBundle.setTimeTaken(executionTime);
            statDataHandler.addResultBundle(resultBundle);
            System.out.println(sql+" | "+ response + ": " + executionTime);
          }
        }
        catch(Exception e)
        {
          e.printStackTrace();
        }
        statDataHandler.signOFF();

    }

}
```

HTTP CONSTRUCTION:

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package com.pmeter.executor;


import com.pmeter.xmlparser.RequestUrl;
import com.pmeter.xmlparser.UrlParameters;
import java.util.Collection;
import com.pmeter.db.DBConnection;
import com.pmeter.util.PMeterUtil;
import com.pmeter.xmlparser.DataHandler;
import com.pmeter.xmlparser.HttpObject;
import java.sql.Connection;

/**
 *
 * @author sony
 */
public class HttpConstruction
{

    public HttpConstruction()
    {
    }

    public void executeHttp(HttpObject httpObject, String outputFileName, int projectID)
    {
       try
       {
          PMeterUtil pMeterUtil = new PMeterUtil();
          Connection    connection    =    pMeterUtil.getConnection("jdbc:mysql://localhost/pmeter",
 "root", "root");
          DataHandler dataHandler = new DataHandler(connection);
          dataHandler.setThreadCound(httpObject.getThreadCount());
          dataHandler.setProjectId(projectID);
          dataHandler.setFileName(outputFileName);

          Collection<RequestUrl> urls = httpObject.getUrlsObject();
          String method = httpObject.getmethod();
          for(int i=0; i<httpObject.getThreadCount(); i++)
          {
             ExecuteHttp es = new ExecuteHttp(dataHandler, httpObject, method);
             es.start();
          }
       }
      catch(Exception e)
      {
         e.printStackTrace();
      }
   }

}
```

STATEMENT CONSTRUCTION:

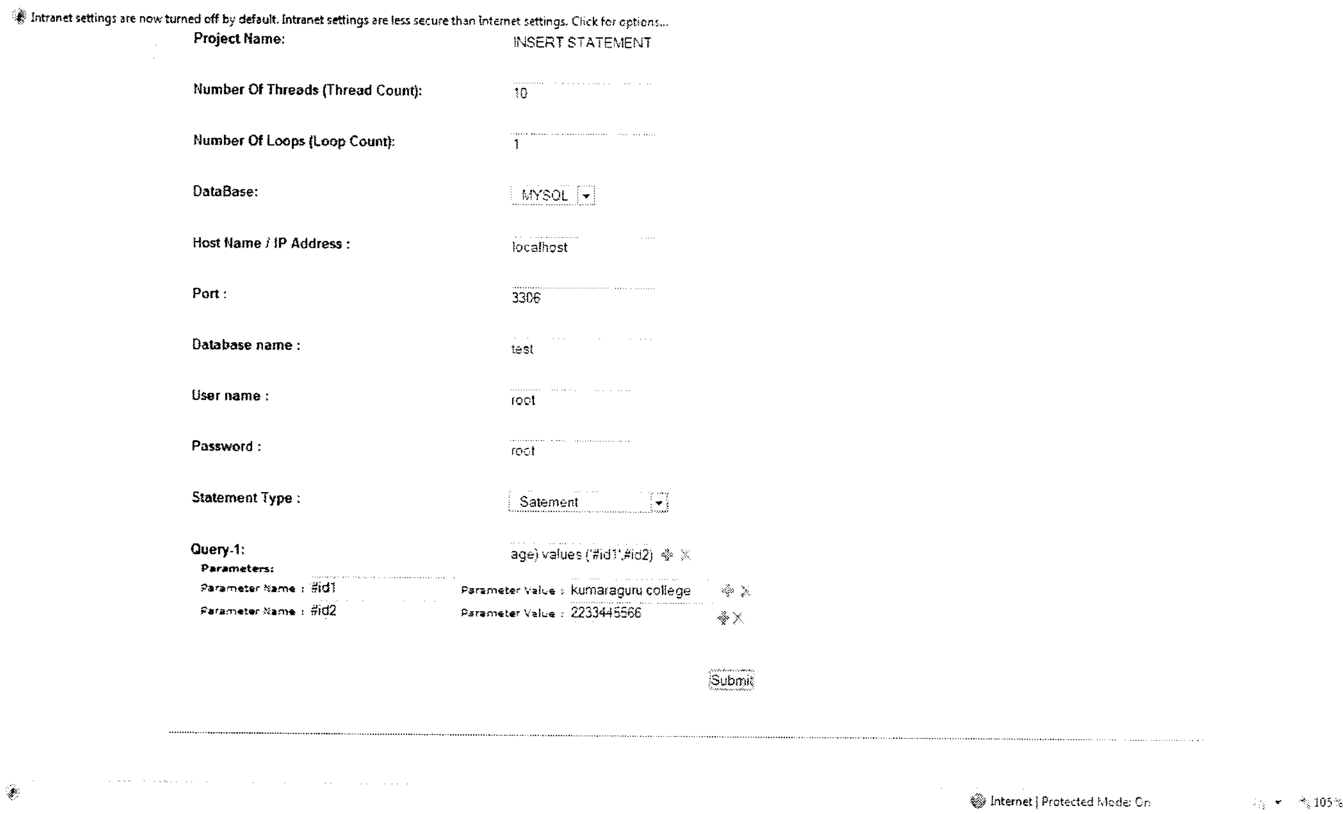
```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package com.pmeter.executor;


import com.pmeter.xmlparser.Queries;
import com.pmeter.xmlparser.QueryParameters;
import java.util.Collection;
import com.pmeter.db.DBConnection;
import com.pmeter.xmlparser.DataHandler;
import com.pmeter.xmlparser.QueryObject;
import java.sql.Connection;
```

```
/**
 *
 * @author sony
 */

public class StatementConstruction
{

   public StatementConstruction()
   {

   }

   public void executeStatement(QueryObject qObject, String outputFileName, int projectID)
   {
      try
      {
         DBConnection dbConnection = new DBConnection();
         Connection    connection    =    dbConnection.getConnection(qObject.getDatabaseType(),
qObject.getHostName(),  qObject.getPort(),  qObject.getDBName(),  qObject.getDBUserName(),
qObject.getDBPassword());

         DataHandler dataHandler = new DataHandler(connection);
         dataHandler.setThreadCound(qObject.getThreadCount());
         dataHandler.setProjectId(projectID);
         dataHandler.setFileName(outputFileName);
         Collection<Queries> queries = qObject.getQueriesObject();
         String statementType = qObject.getStatementType();
         for(int i=0; i<qObject.getThreadCount(); i++)
         {
            ExecuteStatement  es  =  new  ExecuteStatement(dataHandler,  connection,  queries,
statementType);
            es.start();
         }
      }
      catch(Exception e)
      {
         e.printStackTrace();
      }
}
```

6.2 Screenshots

A screenshot is the output of the entire screen in a common bitmap image format such as BMP,PNG or JPEG. This chapter gives a few screenshots of the important functionalities of this applications.

DB INSERT STATEMENT-QUERY EXECUTION



Intranet settings are now turned off by default. Intranet settings are less secure than Internet settings. Click for options...

Project Name:	INSERT STATEMENT
Number Of Threads (Thread Count):	10
Number Of Loops (Loop Count):	1
DataBase:	MYSQL
Host Name / IP Address :	localhost
Port :	3306
Database name :	test
User name :	root
Password :	root
Statement Type :	Satement
Query-1:	age) values ('#id1',#id2)

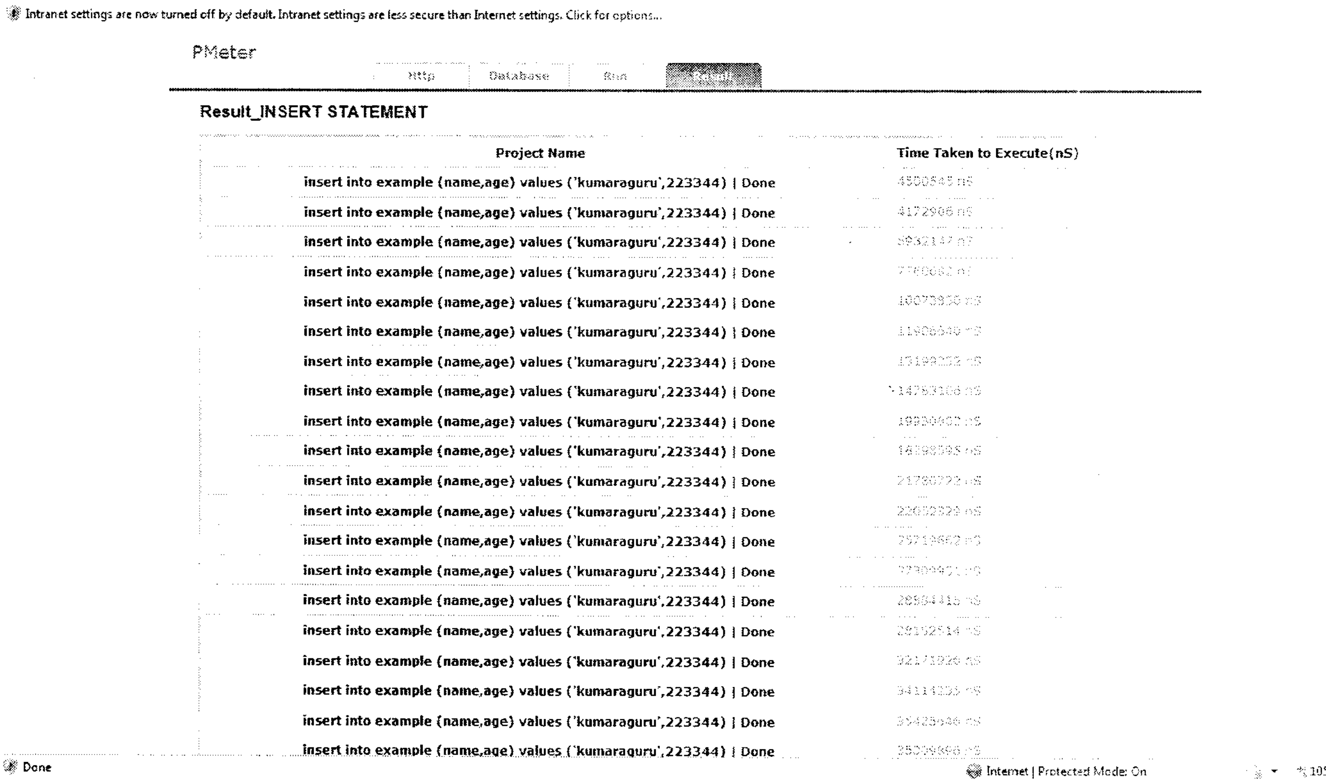
Parameters:

Parameter Name :	#id1	Parameter Value :	kumaraguru college
Parameter Name :	#id2	Parameter Value :	2233445566

Submit

Internet | Protected Mode: On 105%

DB INSERT STATEMENT-RESULT



Intranet settings are now turned off by default. Intranet settings are less secure than Internet settings. Click for options...

PMeter

Http | Database | Run | Result

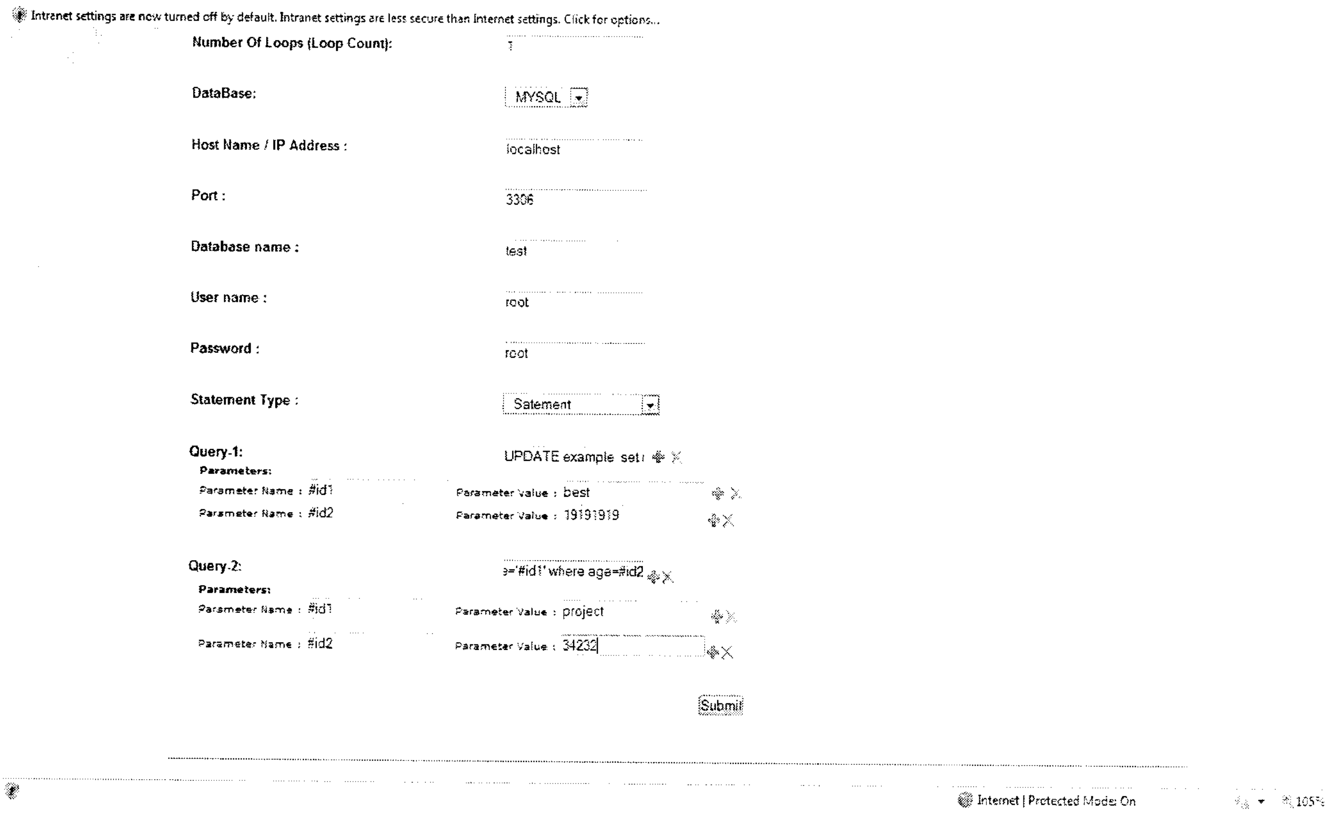
Result_INSERT STATEMENT

Project Name	Time Taken to Execute(nS)
insert into example (name,age) values ('kumaraguru',223344) | Done	4500545 nS
insert into example (name,age) values ('kumaraguru',223344) | Done	4172906 nS
insert into example (name,age) values ('kumaraguru',223344) | Done	5932147 nS
insert into example (name,age) values ('kumaraguru',223344) | Done	7760082 nS
insert into example (name,age) values ('kumaraguru',223344) | Done	10073930 nS
insert into example (name,age) values ('kumaraguru',223344) | Done	11906640 nS
insert into example (name,age) values ('kumaraguru',223344) | Done	13199232 nS
insert into example (name,age) values ('kumaraguru',223344) | Done	14763106 nS
insert into example (name,age) values ('kumaraguru',223344) | Done	19930402 nS
insert into example (name,age) values ('kumaraguru',223344) | Done	16898595 nS
insert into example (name,age) values ('kumaraguru',223344) | Done	21780722 nS
insert into example (name,age) values ('kumaraguru',223344) | Done	22052329 nS
insert into example (name,age) values ('kumaraguru',223344) | Done	25719662 nS
insert into example (name,age) values ('kumaraguru',223344) | Done	27309901 nS
insert into example (name,age) values ('kumaraguru',223344) | Done	28894416 nS
insert into example (name,age) values ('kumaraguru',223344) | Done	29152514 nS
insert into example (name,age) values ('kumaraguru',223344) | Done	32171926 nS
insert into example (name,age) values ('kumaraguru',223344) | Done	34114235 nS
insert into example (name,age) values ('kumaraguru',223344) | Done	35425646 nS
insert into example (name,age) values ('kumaraguru',223344) | Done	35039896 nS

Done

Internet | Protected Mode: On

DB UPDATE STATEMENT-QUERY EXECUTION



Intranet settings are now turned off by default. Intranet settings are less secure than Internet settings. Click for options...

Number Of Loops (Loop Count):	1
DataBase:	MYSQL
Host Name / IP Address :	localhost
Port :	3306
Database name :	test
User name :	root
Password :	root
Statement Type :	Satement

Query-1: UPDATE example set

Parameters:

Parameter Name : #id1	Parameter Value : best
Parameter Name : #id2	Parameter Value : 19191919

Query-2: ='#id1' where age=#id2

Parameters:

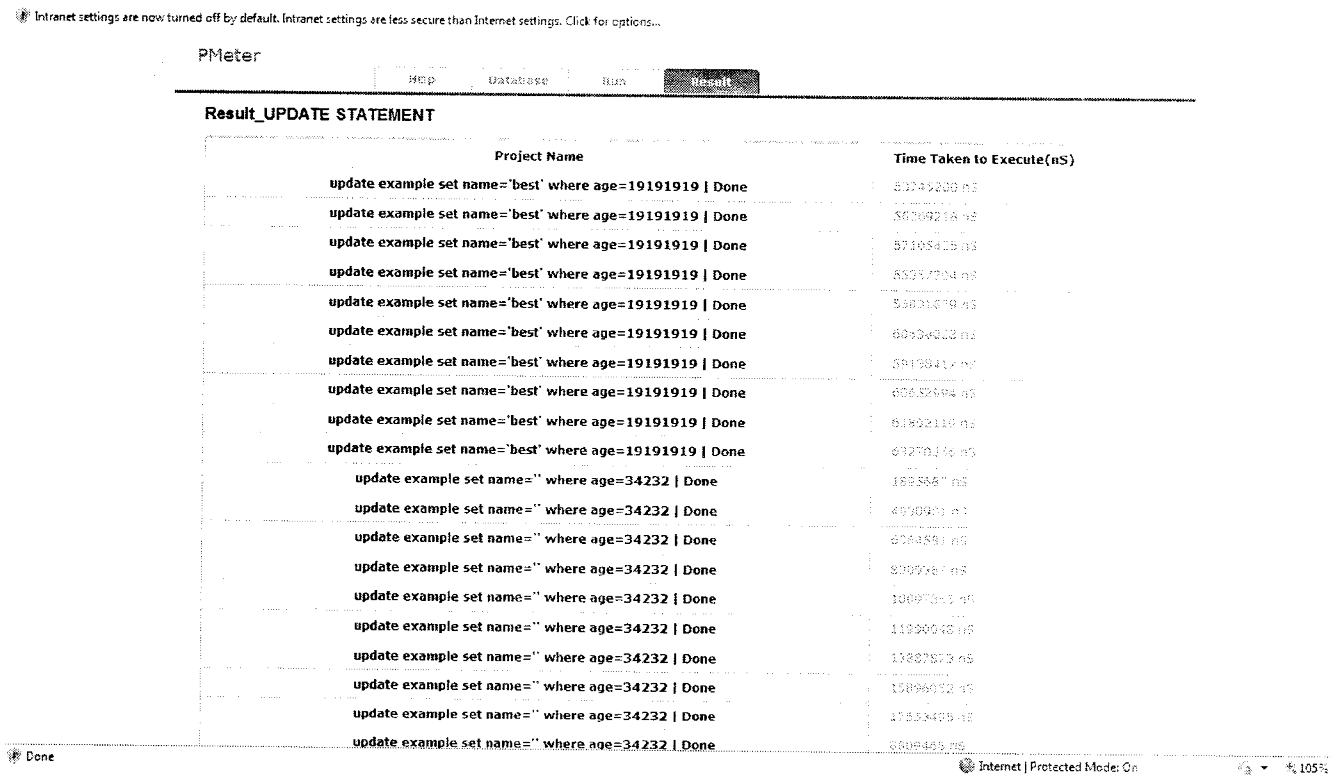
Parameter Name : #id1	Parameter Value : project
Parameter Name : #id2	Parameter Value : 34232

Submit

Internet | Protected Mode: On

105%

DB UPDATE STATEMENT-RESULT



Intranet settings are now turned off by default. Intranet settings are less secure than Internet settings. Click for options...

PMeter

Help | Database | Run | Result

Result_UPDATE STATEMENT

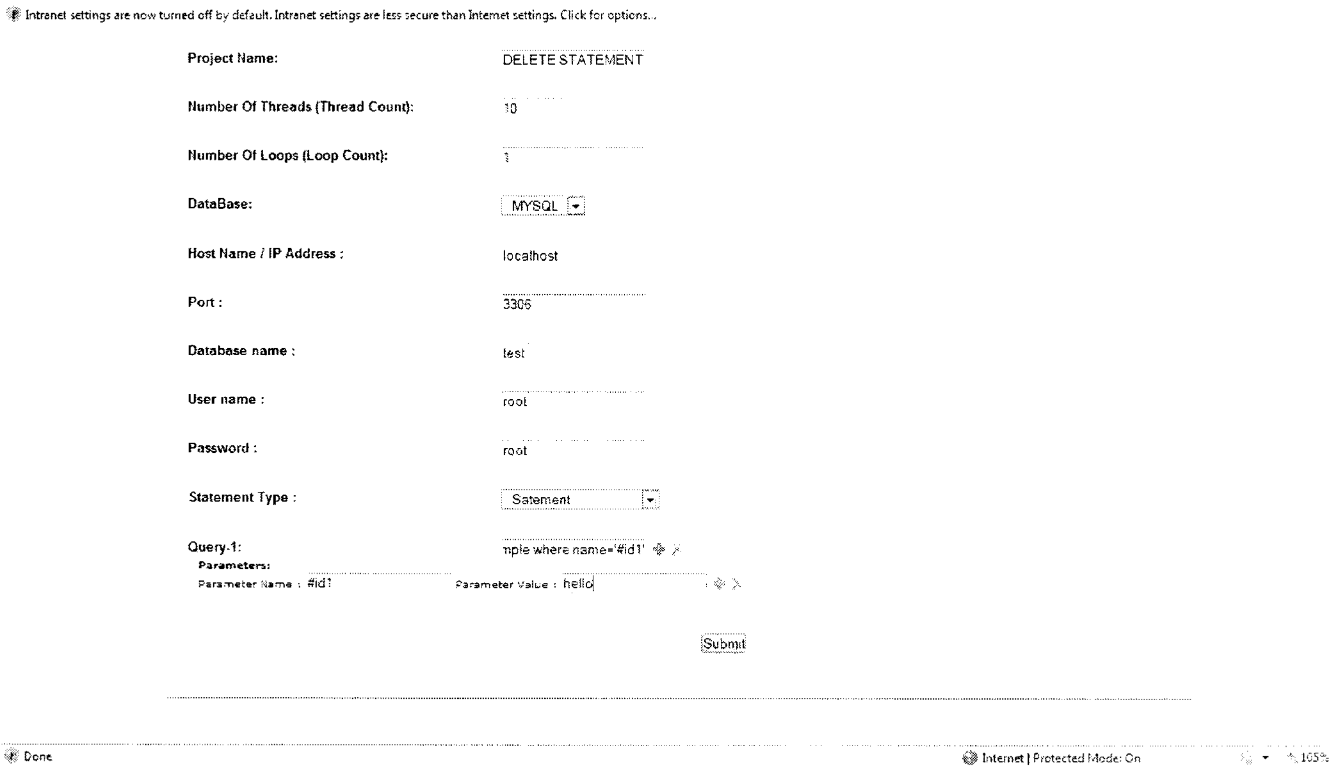
Project Name	Time Taken to Execute(nS)
update example set name='best' where age=19191919 | Done	53245200 nS
update example set name='best' where age=19191919 | Done	56369218 nS
update example set name='best' where age=19191919 | Done	57105403 nS
update example set name='best' where age=19191919 | Done	55357204 nS
update example set name='best' where age=19191919 | Done	53831679 nS
update example set name='best' where age=19191919 | Done	60534032 nS
update example set name='best' where age=19191919 | Done	59199412 nS
update example set name='best' where age=19191919 | Done	60652694 nS
update example set name='best' where age=19191919 | Done	61802110 nS
update example set name='best' where age=19191919 | Done	63270136 nS
update example set name='' where age=34232 | Done	1895687 nS
update example set name='' where age=34232 | Done	4000905 nS
update example set name='' where age=34232 | Done	6744581 nS
update example set name='' where age=34232 | Done	8209681 nS
update example set name='' where age=34232 | Done	10097345 nS
update example set name='' where age=34232 | Done	11900048 nS
update example set name='' where age=34232 | Done	13887873 nS
update example set name='' where age=34232 | Done	15896052 nS
update example set name='' where age=34232 | Done	17533405 nS
update example set name='' where age=34232 | Done	6809465 nS

Done

Internet | Protected Mode: On

105%

DB DELETE STATEMENT-QUERY EXECUTION



Intranet settings are now turned off by default. Intranet settings are less secure than Internet settings. Click for options...

Project Name:	DELETE STATEMENT
Number Of Threads (Thread Count):	10
Number Of Loops (Loop Count):	1
DataBase:	MYSQL
Host Name / IP Address :	localhost
Port :	3306
Database name :	test
User name :	root
Password :	root
Statement Type :	Satement
Query-1:	mple where name='#id1'

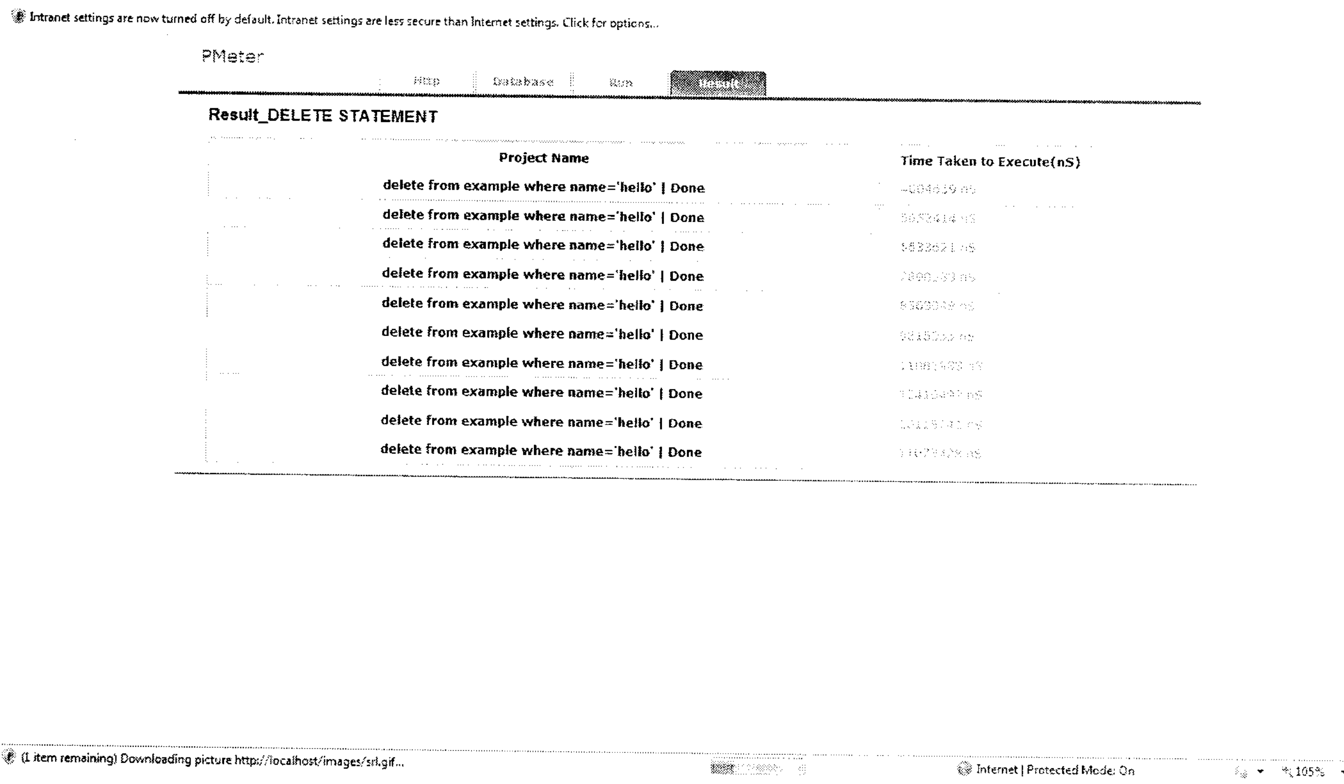
Parameters:

Parameter Name : #id1 Parameter Value : hello

Submit

Done Internet | Protected Mode: On 105%

DB DELETE STATEMENT-RESULT



Intranet settings are now turned off by default. Intranet settings are less secure than Internet settings. Click for options...

PMeter

Http | Database | Run | Result

Result_DELETE STATEMENT

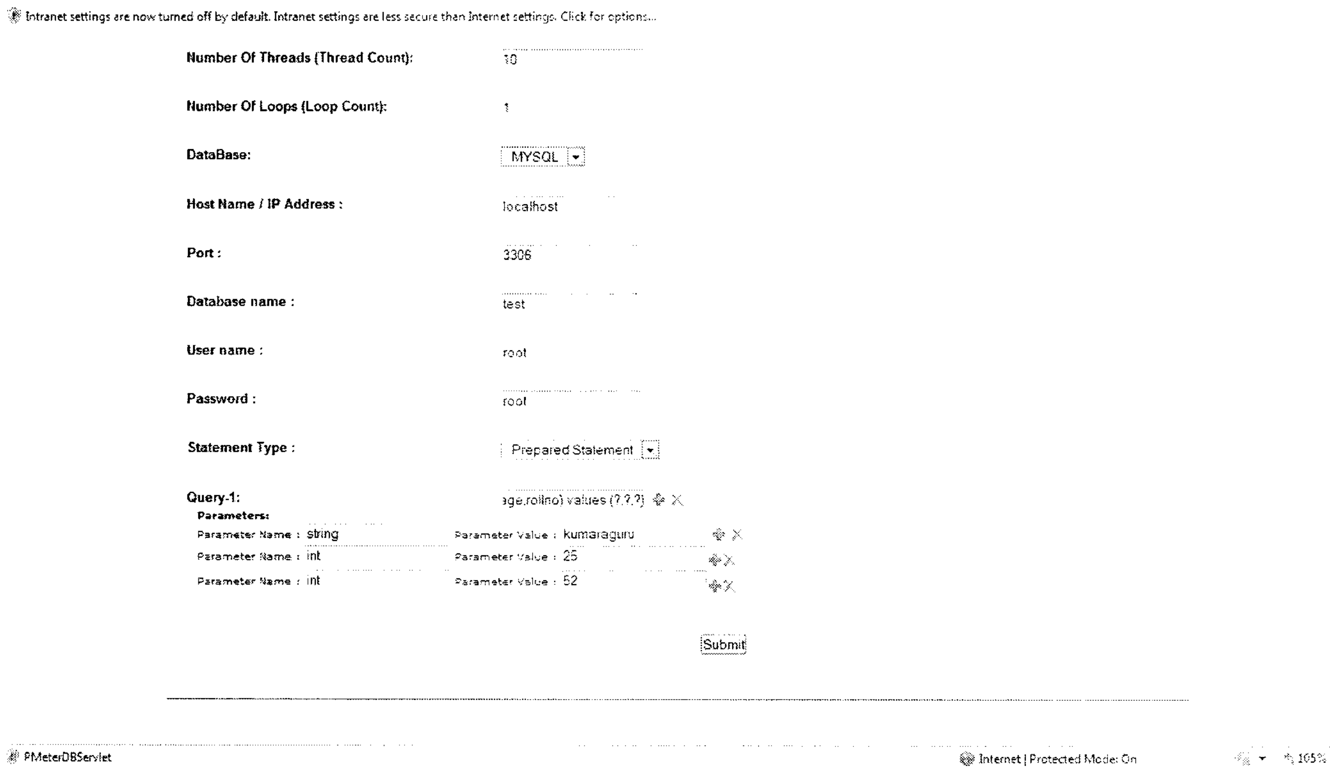
Project Name	Time Taken to Execute(nS)
delete from example where name='hello' | Done	4004659 nS
delete from example where name='hello' | Done	5652414 nS
delete from example where name='hello' | Done	5833621 nS
delete from example where name='hello' | Done	7800293 nS
delete from example where name='hello' | Done	8300048 nS
delete from example where name='hello' | Done	9215535 nS
delete from example where name='hello' | Done	11081493 nS
delete from example where name='hello' | Done	10413497 nS
delete from example where name='hello' | Done	10129141 nS
delete from example where name='hello' | Done	11679428 nS

(1 item remaining) Downloading picture http://localhost/images/srl.gif...

Internet | Protected Mode: On

105%

DB INSERT PREPARED STATEMENT-QUERY EXECUTION



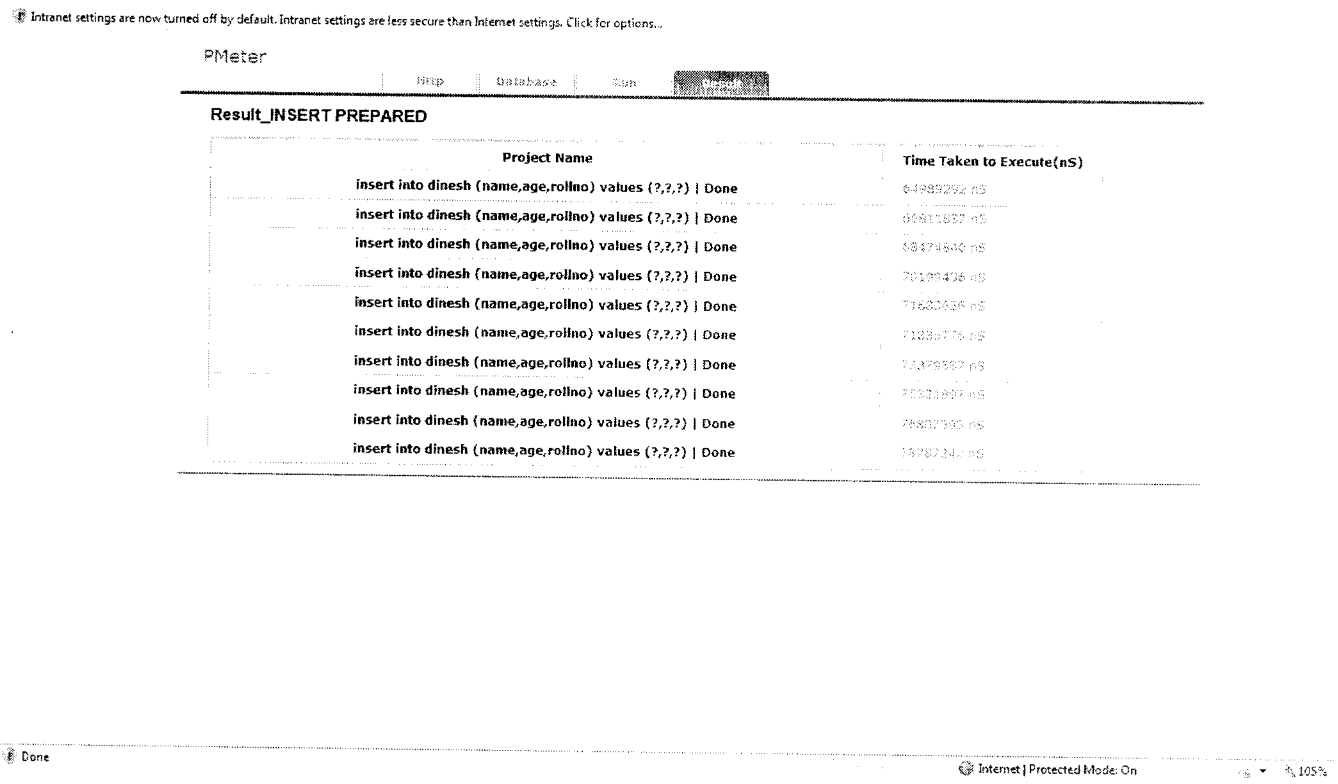
Intranet settings are now turned off by default. Intranet settings are less secure than Internet settings. Click for options...

Number Of Threads (Thread Count):	10
Number Of Loops (Loop Count):	1
DataBase:	MYSQL
Host Name / IP Address :	localhost
Port :	3306
Database name :	test
User name :	root
Password :	root
Statement Type :	Prepared Statement
Query-1:	age,rollno) values (?,?,?)

Parameters:

Parameter Name :	string	Parameter Value :	kumaraguru
Parameter Name :	int	Parameter Value :	25
Parameter Name :	int	Parameter Value :	52

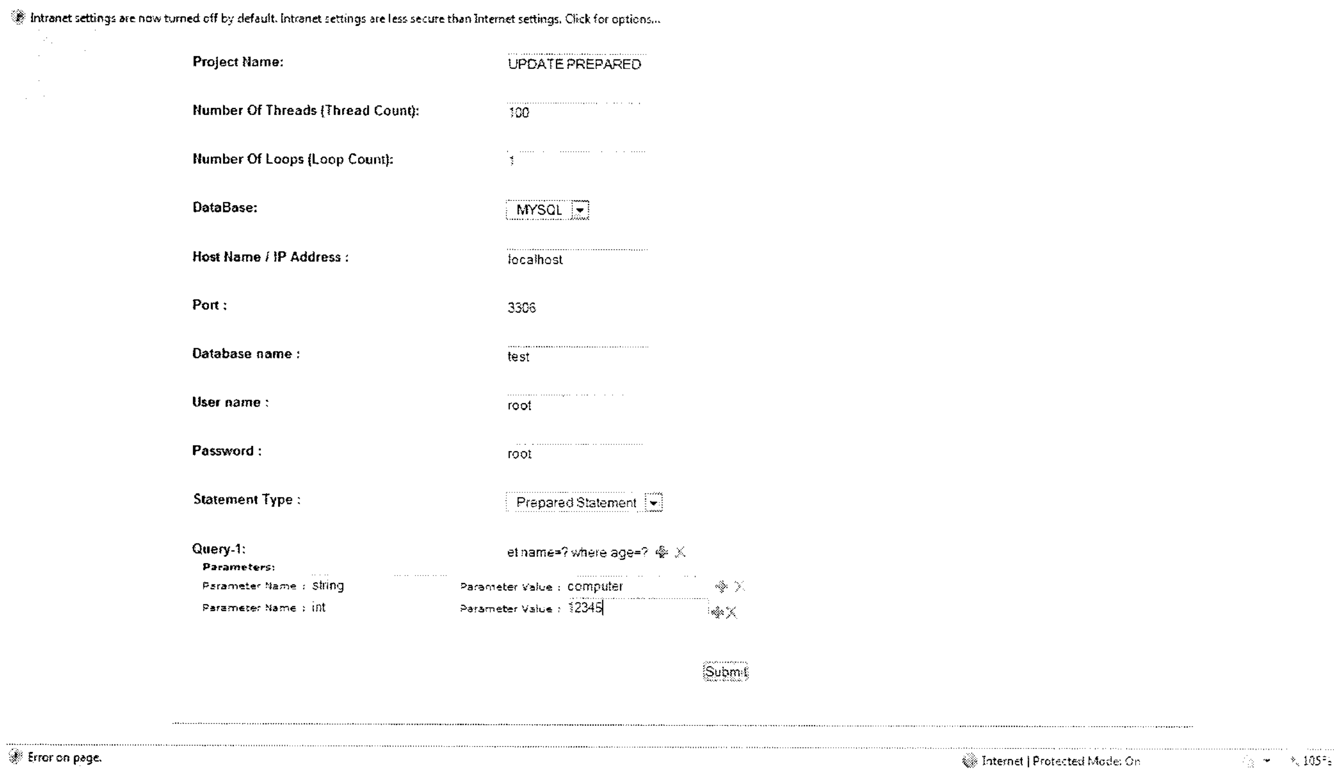
DB INSERT PREPARED STATEMENT-RESULT


Intranet settings are now turned off by default. Intranet settings are less secure than Internet settings. Click for options...
PMeter
Http
Database
Run
Result
Done
Internet | Protected Mode: On
105%

Result_INSERT PREPARED

Project Name	Time Taken to Execute(nS)
insert into dinesh (name,age,rollno) values (?,?,?) | Done	64989292 nS
insert into dinesh (name,age,rollno) values (?,?,?) | Done	66811897 nS
insert into dinesh (name,age,rollno) values (?,?,?) | Done	68474930 nS
insert into dinesh (name,age,rollno) values (?,?,?) | Done	70199406 nS
insert into dinesh (name,age,rollno) values (?,?,?) | Done	71680658 nS
insert into dinesh (name,age,rollno) values (?,?,?) | Done	71885776 nS
insert into dinesh (name,age,rollno) values (?,?,?) | Done	72379597 nS
insert into dinesh (name,age,rollno) values (?,?,?) | Done	75321897 nS
insert into dinesh (name,age,rollno) values (?,?,?) | Done	76807390 nS
insert into dinesh (name,age,rollno) values (?,?,?) | Done	15787241 nS

DB UPDATE PREPARED STATEMENT-QUERY EXECUTION



Intranet settings are now turned off by default. Intranet settings are less secure than Internet settings. Click for options...

Project Name:	UPDATE PREPARED
Number Of Threads (Thread Count):	100
Number Of Loops (Loop Count):	1
DataBase:	MYSQL
Host Name / IP Address :	localhost
Port :	3306
Database name :	test
User name :	root
Password :	root
Statement Type :	Prepared Statement
Query-1:	et name=? where age=?

Parameters:

Parameter Name :	String	Parameter Value :	computer
Parameter Name :	int	Parameter Value :	12345

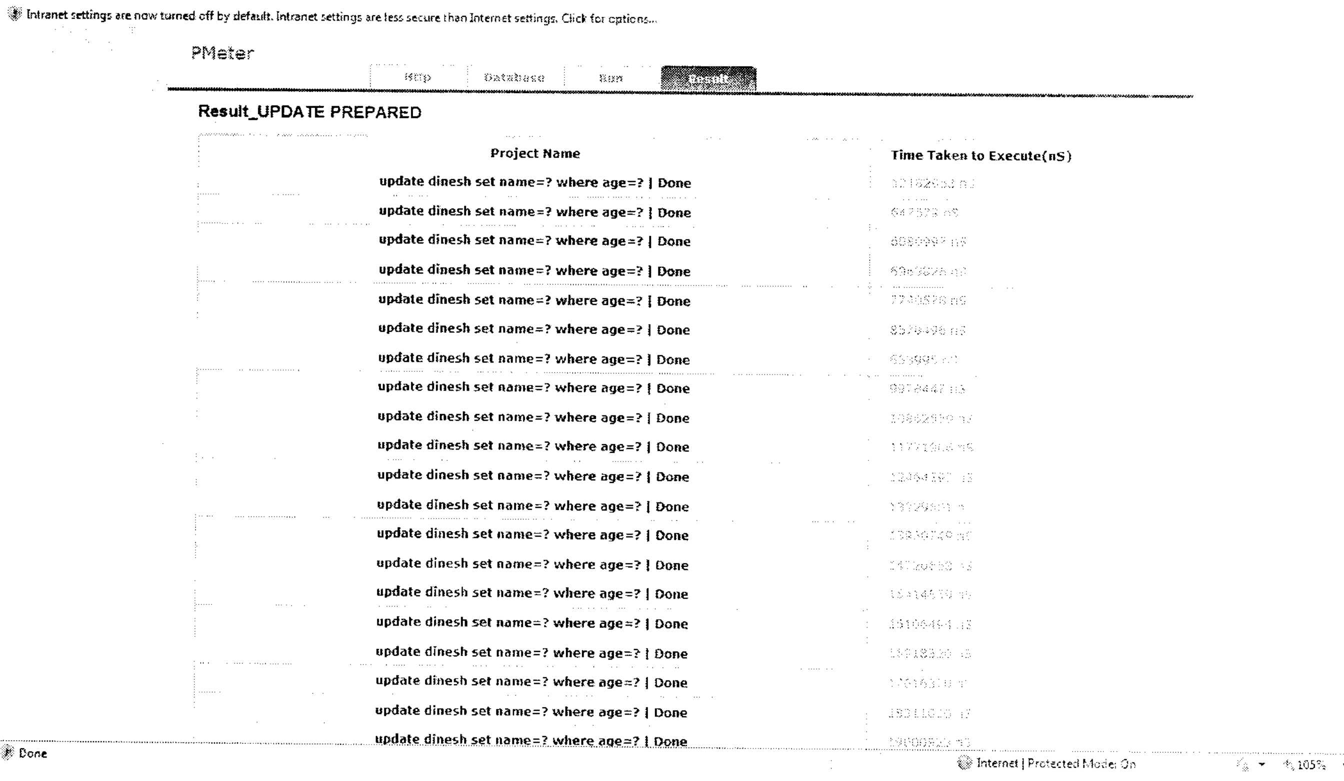
Submit

Error on page.

Internet \| Protected Mode: On

105%

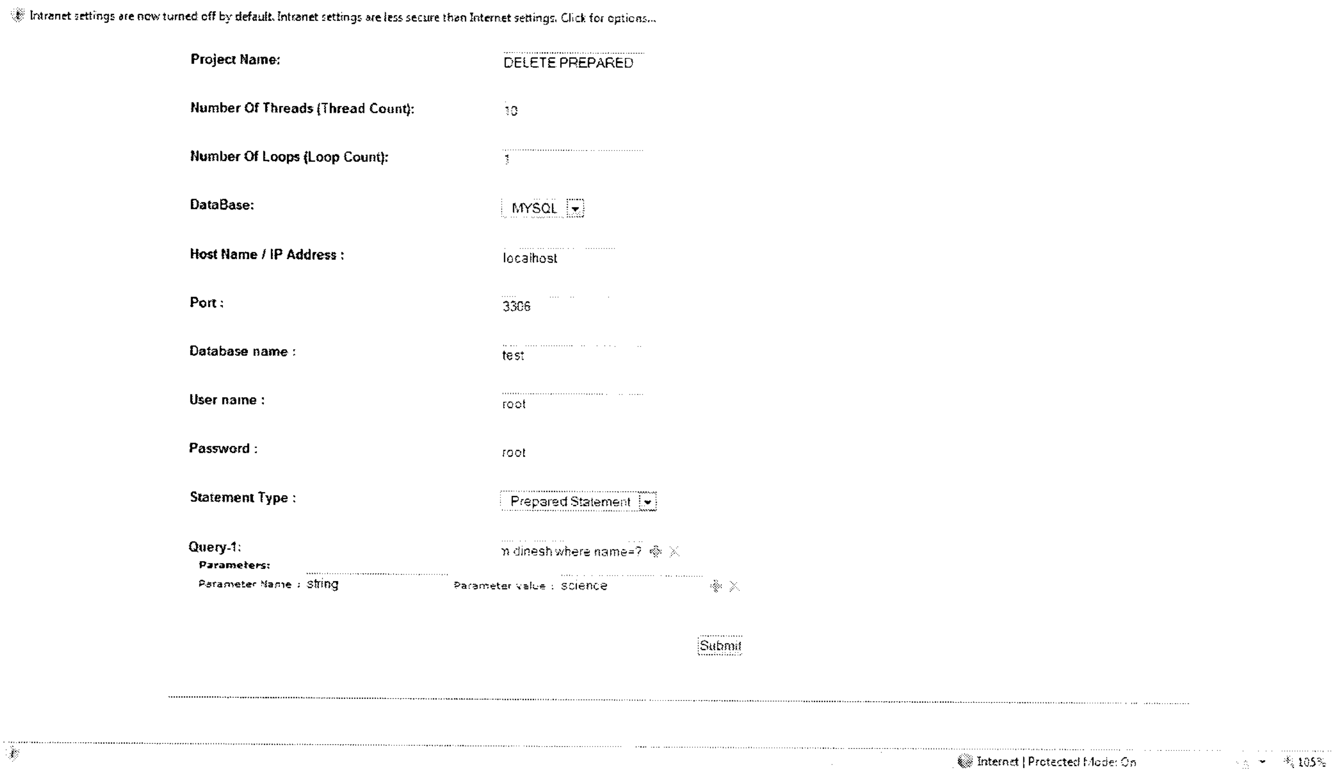
DB UPDATE PREPARED STATEMENT-RESULT


Intranet settings are now turned off by default. Intranet settings are less secure than Internet settings. Click for options...
PMeter
Http
Database
Run
Result
Done
Internet | Protected Mode: On
105%

Result_UPDATE PREPARED

Project Name	Time Taken to Execute(nS)
update dinesh set name=? where age=? | Done	[illegible]
update dinesh set name=? where age=? | Done	647573 nS
update dinesh set name=? where age=? | Done	6080997 nS
update dinesh set name=? where age=? | Done	6963626 nS
update dinesh set name=? where age=? | Done	7740576 nS
update dinesh set name=? where age=? | Done	8576498 nS
update dinesh set name=? where age=? | Done	[illegible]
update dinesh set name=? where age=? | Done	9978447 nS
update dinesh set name=? where age=? | Done	10862950 nS
update dinesh set name=? where age=? | Done	11771814 nS
update dinesh set name=? where age=? | Done	12464397 nS
update dinesh set name=? where age=? | Done	[illegible]
update dinesh set name=? where age=? | Done	13836749 nS
update dinesh set name=? where age=? | Done	14726850 nS
update dinesh set name=? where age=? | Done	15414570 nS
update dinesh set name=? where age=? | Done	16106464 nS
update dinesh set name=? where age=? | Done	16718320 nS
update dinesh set name=? where age=? | Done	[illegible]
update dinesh set name=? where age=? | Done	18311020 nS
update dinesh set name=? where age=? | Done	19008822 nS

DB DELETE PREPARED STATEMENT-QUERY EXECUTION



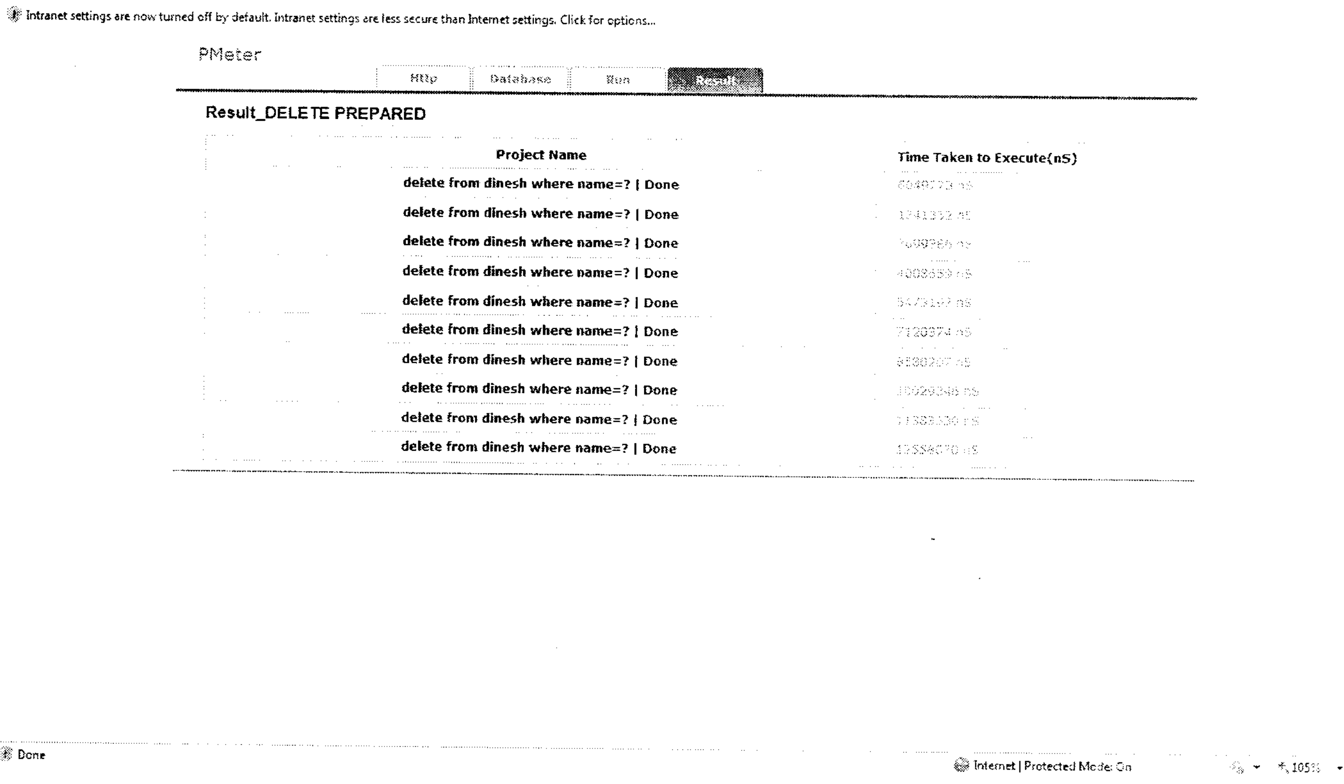
Intranet settings are now turned off by default. Intranet settings are less secure than Internet settings. Click for options...

Project Name:	DELETE PREPARED
Number Of Threads (Thread Count):	10
Number Of Loops (Loop Count):	1
DataBase:	MYSQL
Host Name / IP Address :	localhost
Port :	3306
Database name :	test
User name :	root
Password :	root
Statement Type :	Prepared Statement
Query-1:	n dinesh where name=?

Parameters:

Parameter Name : String Parameter Value : science

DB DELETE PREPARED STATEMENT-RESULT

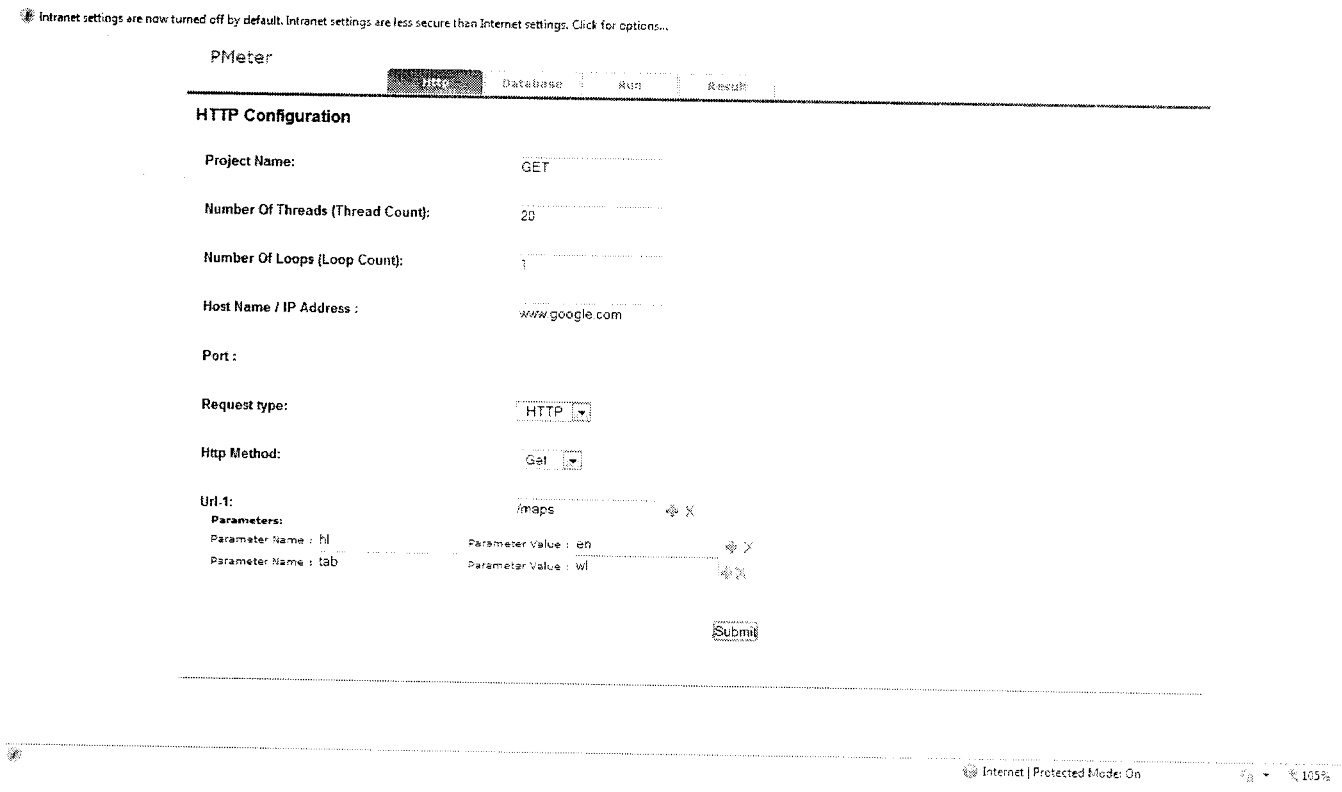


Intranet settings are now turned off by default. Intranet settings are less secure than Internet settings. Click for options...

Result_DELETE PREPARED

Project Name	Time Taken to Execute(nS)
delete from dinesh where name=? | Done	6049773 nS
delete from dinesh where name=? | Done	1741332 nS
delete from dinesh where name=? | Done	[illegible]699786 nS
delete from dinesh where name=? | Done	4008659 nS
delete from dinesh where name=? | Done	5473197 nS
delete from dinesh where name=? | Done	7120374 nS
delete from dinesh where name=? | Done	8580207 nS
delete from dinesh where name=? | Done	10029348 nS
delete from dinesh where name=? | Done	11383336 nS
delete from dinesh where name=? | Done	12558070 nS

HTTP GET METHOD-EXECUTION



Intranet settings are now turned off by default. Intranet settings are less secure than Internet settings. Click for options...

PMeter

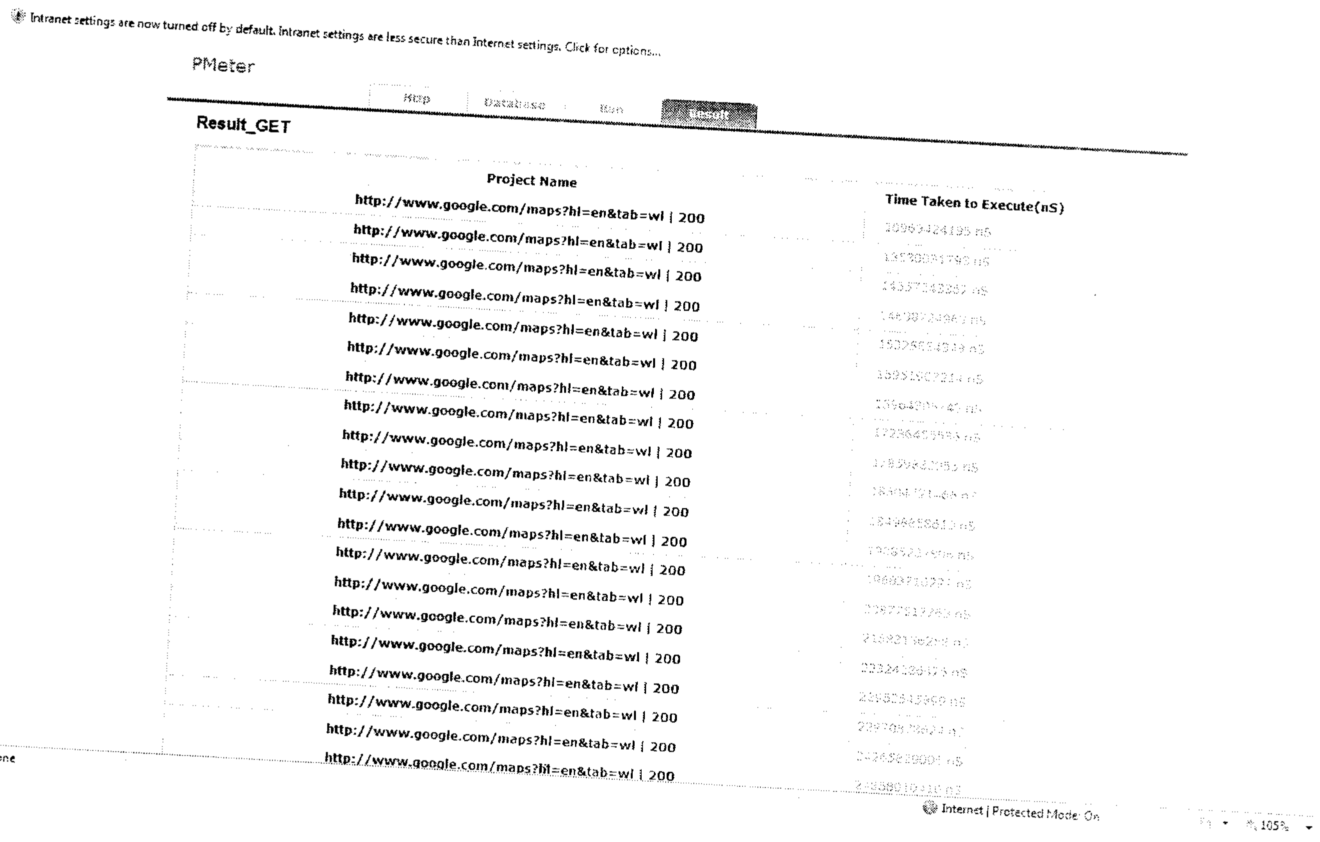
HTTP Configuration

Project Name:	GET
Number Of Threads (Thread Count):	20
Number Of Loops (Loop Count):	1
Host Name / IP Address :	www.google.com
Port :	
Request type:	HTTP
Http Method:	Get
Url-1:	/maps

Parameters:

Parameter Name : hl	Parameter Value : en
Parameter Name : tab	Parameter Value : wl

HTTP GET METHOD-RESULT



Intranet settings are now turned off by default. Intranet settings are less secure than Internet settings. Click for options...

PMeter

Http | Database | Run | Result

Result_GET

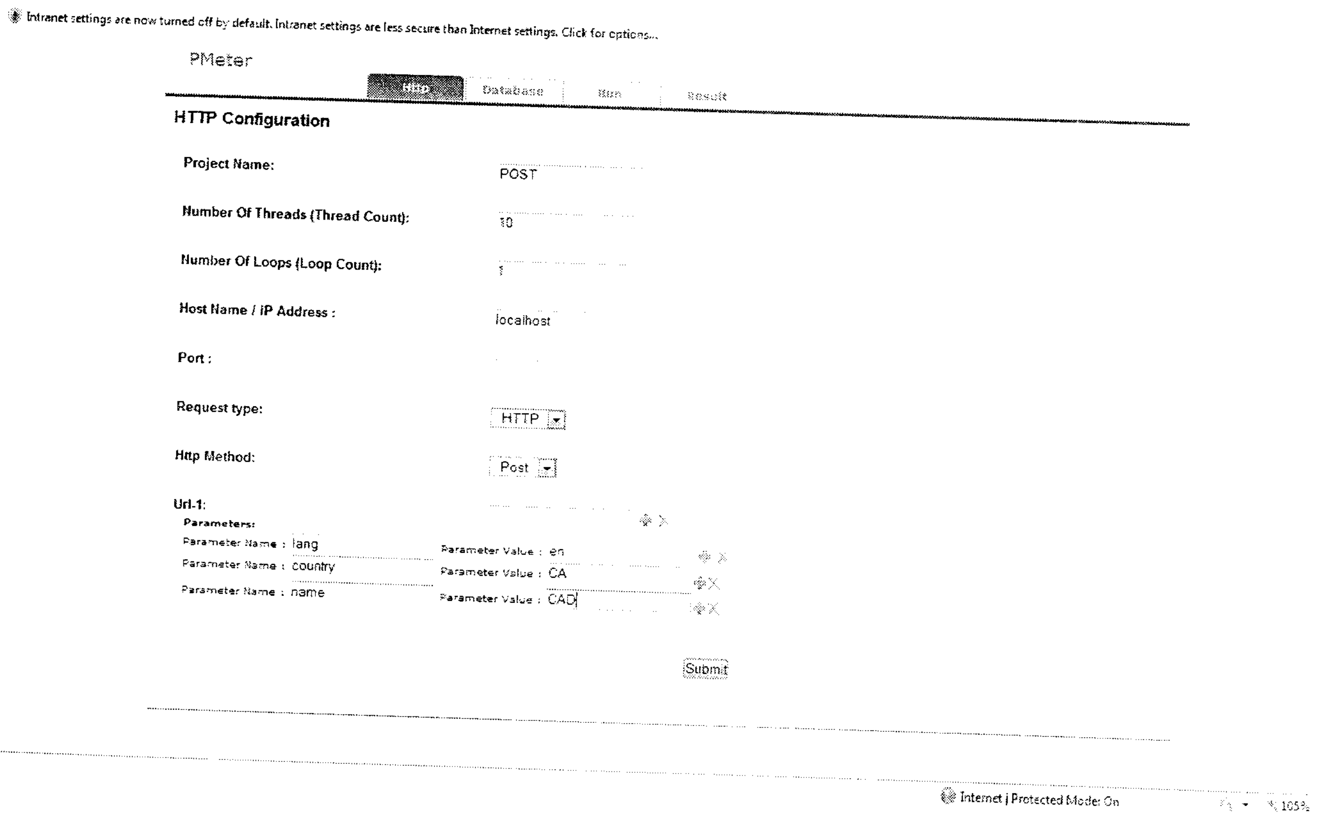
Project Name	Time Taken to Execute(nS)
http://www.google.com/maps?hl=en&tab=wl | 200	10963424195 nS
http://www.google.com/maps?hl=en&tab=wl | 200	13530081790 nS
http://www.google.com/maps?hl=en&tab=wl | 200	14357342282 nS
http://www.google.com/maps?hl=en&tab=wl | 200	14630724967 nS
http://www.google.com/maps?hl=en&tab=wl | 200	15026654939 nS
http://www.google.com/maps?hl=en&tab=wl | 200	15951507214 nS
http://www.google.com/maps?hl=en&tab=wl | 200	[illegible]
http://www.google.com/maps?hl=en&tab=wl | 200	17236405553 nS
http://www.google.com/maps?hl=en&tab=wl | 200	17859842753 nS
http://www.google.com/maps?hl=en&tab=wl | 200	[illegible]
http://www.google.com/maps?hl=en&tab=wl | 200	18496658610 nS
http://www.google.com/maps?hl=en&tab=wl | 200	[illegible]
http://www.google.com/maps?hl=en&tab=wl | 200	19680710291 nS
http://www.google.com/maps?hl=en&tab=wl | 200	20977317762 nS
http://www.google.com/maps?hl=en&tab=wl | 200	[illegible]
http://www.google.com/maps?hl=en&tab=wl | 200	22324126173 nS
http://www.google.com/maps?hl=en&tab=wl | 200	22582545990 nS
http://www.google.com/maps?hl=en&tab=wl | 200	[illegible]
http://www.google.com/maps?hl=en&tab=wl | 200	24265818004 nS
http://www.google.com/maps?hl=en&tab=wl | 200	24580102410 nS

Done

Internet | Protected Mode: On

105%

HTTP POST METHOD-EXECUTION



Intranet settings are now turned off by default. Intranet settings are less secure than Internet settings. Click for options...

PMeter

Http | Database | Run | Result

HTTP Configuration

Project Name:	POST
Number Of Threads (Thread Count):	10
Number Of Loops (Loop Count):	1
Host Name / IP Address :	localhost
Port :	
Request type:	HTTP
Http Method:	Post

Url-1:

Parameters:

Parameter Name : lang	Parameter Value : en
Parameter Name : country	Parameter Value : CA
Parameter Name : name	Parameter Value : CAD

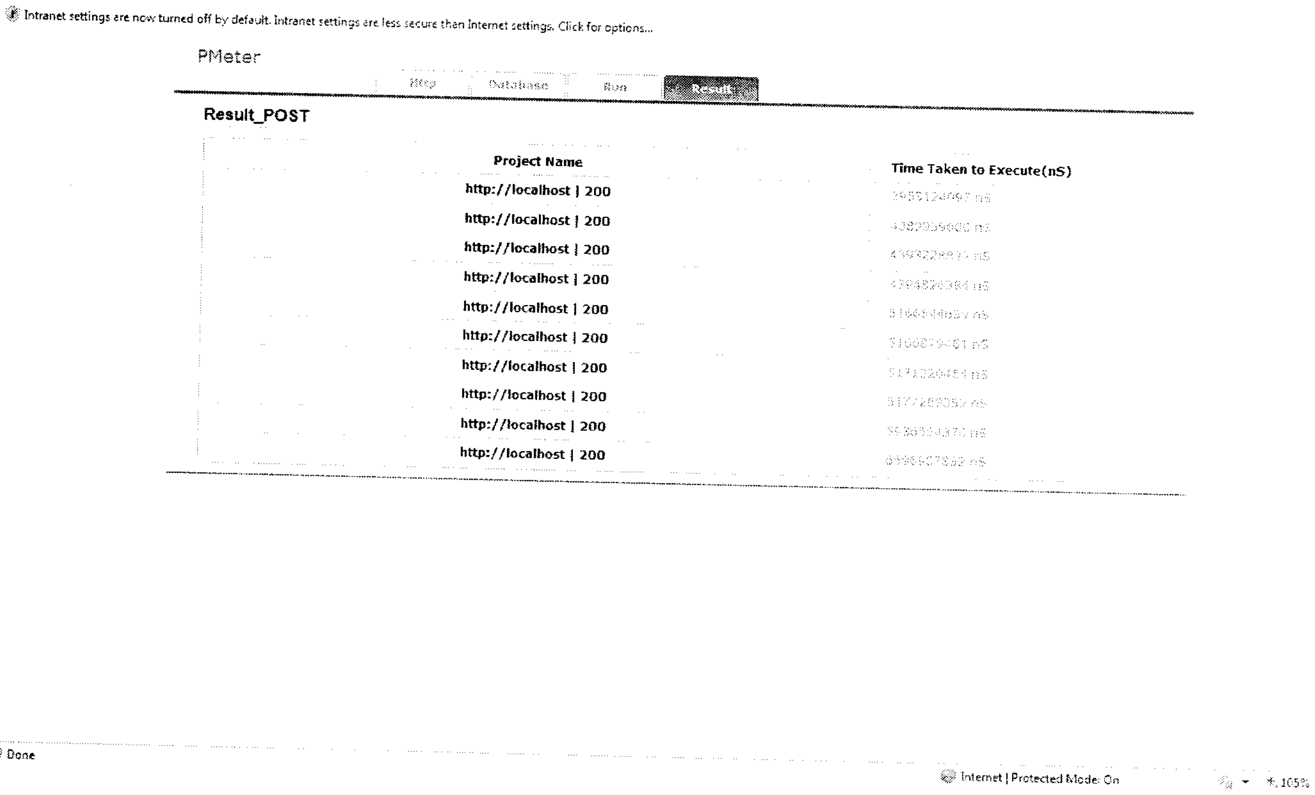
Submit

Internet | Protected Mode: On

105%

HTTP POST METHOD-RESULT

P-3615

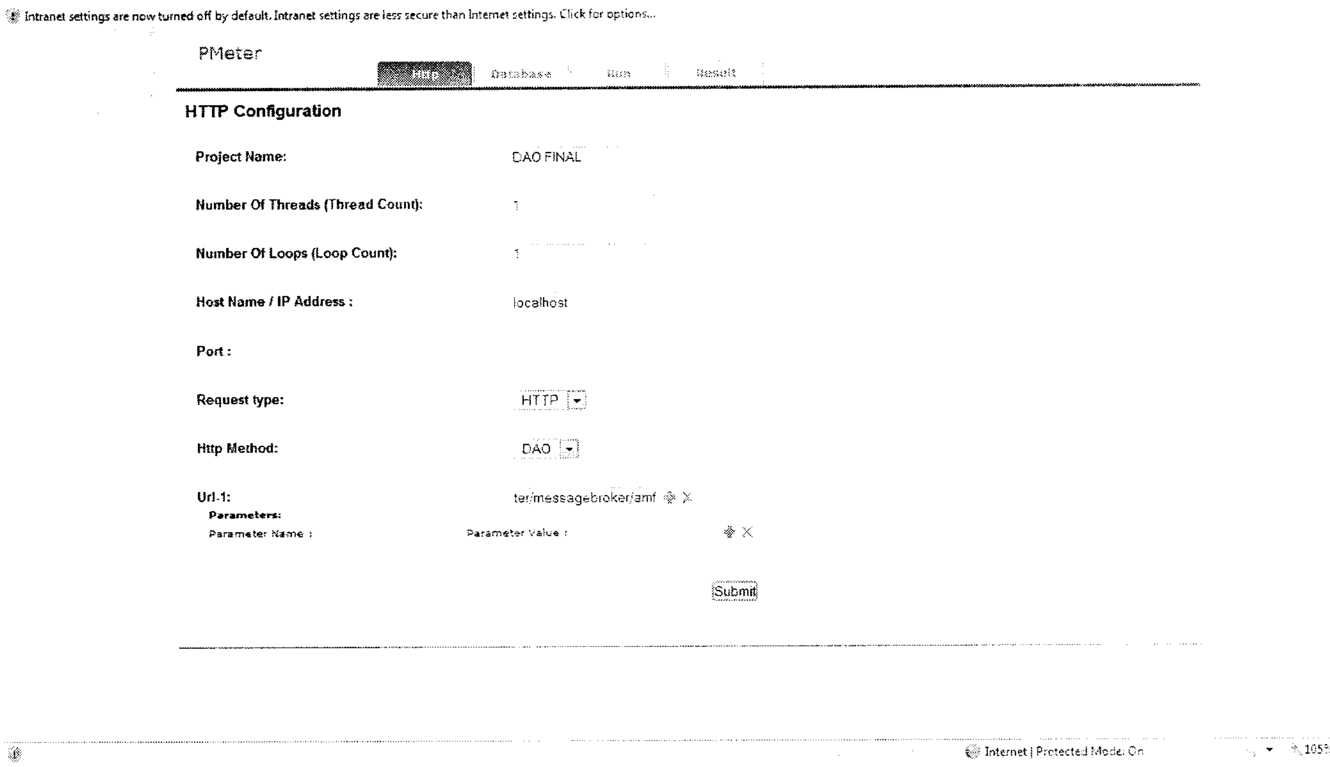


Intranet settings are now turned off by default. Intranet settings are less secure than Internet settings. Click for options...
PMeter
Http
Database
Run
Result

Result_POST

Project Name	Time Taken to Execute(nS)
http://localhost | 200	2955124097 nS
http://localhost | 200	4382959600 nS
http://localhost | 200	4399228972 nS
http://localhost | 200	4394826364 nS
http://localhost | 200	5166844639 nS
http://localhost | 200	5166879461 nS
http://localhost | 200	5171320461 nS
http://localhost | 200	5177289052 nS
http://localhost | 200	5626554370 nS
http://localhost | 200	6398407832 nS

DAO MODULE-EXECUTION (WITHOUT ERROR)

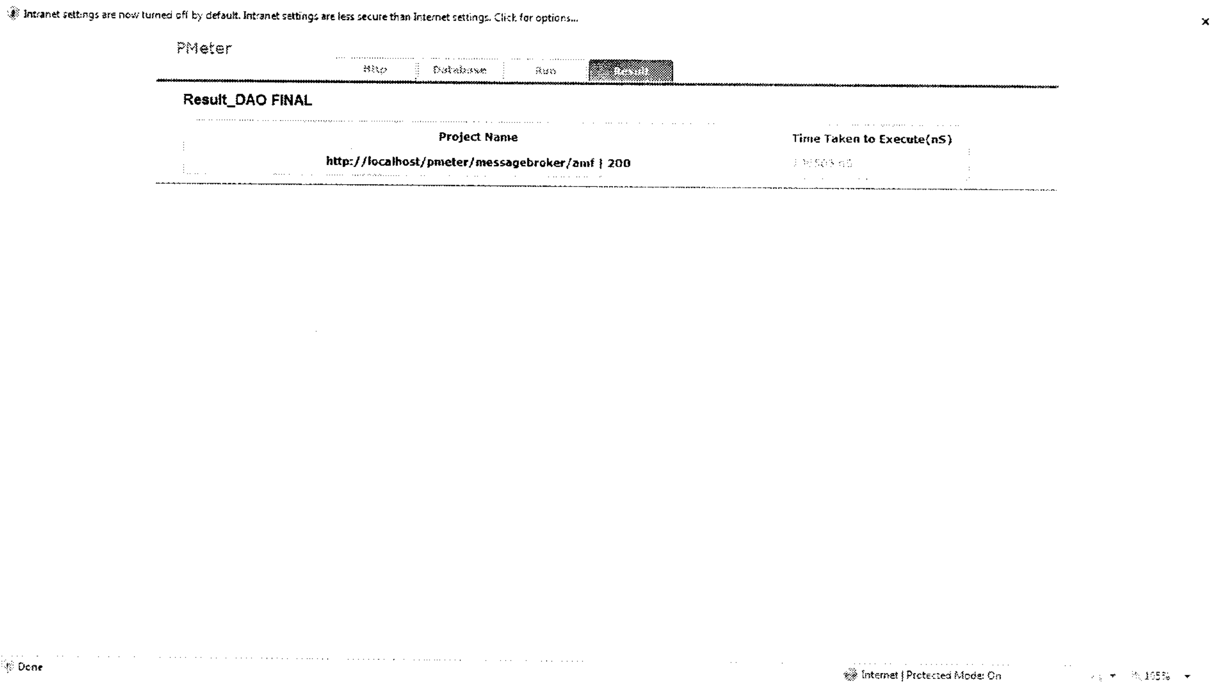

Intranet settings are now turned off by default. Intranet settings are less secure than Internet settings. Click for options...

HTTP Configuration

Project Name:	DAO FINAL
Number Of Threads (Thread Count):	1
Number Of Loops (Loop Count):	1
Host Name / IP Address :	localhost
Port :	
Request type:	HTTP
Http Method:	DAO
Url-1:	ter/messagebroker/amf

Parameters:
Parameter Name : Parameter Value :

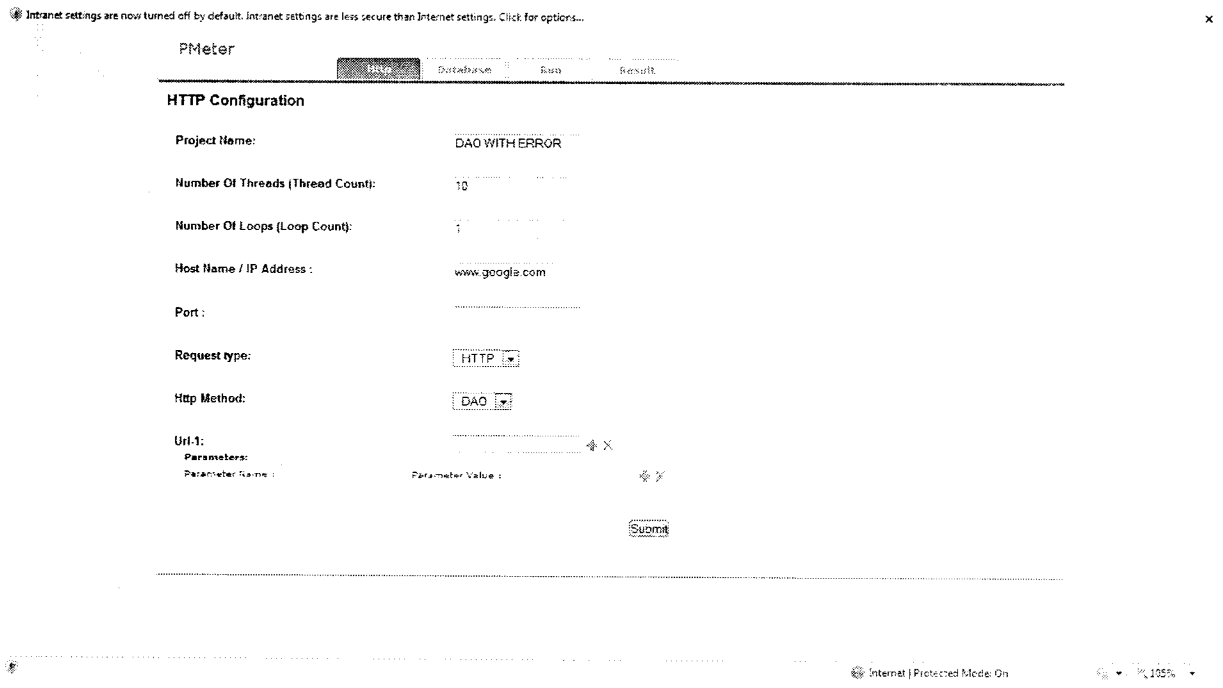
DAO MODULE-RESULT (WITHOUT ERROR)


Intranet settings are now turned off by default. Intranet settings are less secure than Internet settings. Click for options...
PMeter
Http
Database
Run
Result
Done
Internet | Protected Mode: On
105%

Result_DAO FINAL

Project Name	Time Taken to Execute(nS)
http://localhost/pmeter/messagebroker/amf | 200	[illegible] nS

DAO MODULE-EXECUTION (WITH ERROR)



Intranet settings are now turned off by default. Intranet settings are less secure than Internet settings. Click for options...

PMeter

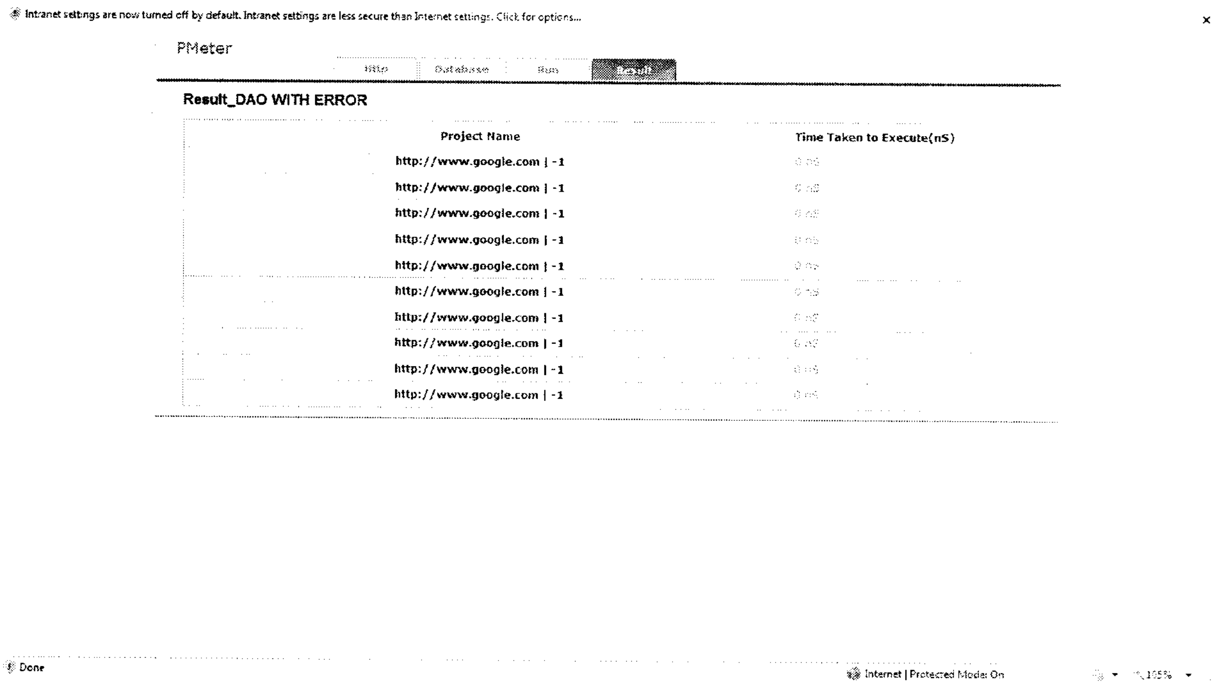
HTTP Configuration

Project Name:	DAO WITH ERROR
Number Of Threads (Thread Count):	10
Number Of Loops (Loop Count):	1
Host Name / IP Address :	www.google.com
Port :	
Request type:	HTTP
Http Method:	DAO
Url-1:	

Parameters:

Parameter Name : Parameter Value :

DAO MODULE-RESULT (WITH ERROR)



Intranet settings are now turned off by default. Intranet settings are less secure than Internet settings. Click for options...

Result_DAO WITH ERROR

Project Name	Time Taken to Execute(nS)
http://www.google.com | -1	0 nS
http://www.google.com | -1	0 nS
http://www.google.com | -1	0 nS
http://www.google.com | -1	0 nS
http://www.google.com | -1	0 nS
http://www.google.com | -1	0 nS
http://www.google.com | -1	0 nS
http://www.google.com | -1	0 nS
http://www.google.com | -1	0 nS
http://www.google.com | -1	0 nS

CHAPTER - 7

7.1 SUMMARY

The performance test plan can be executed and the result for the project is stored in XML and can be viewed in the table format as show in the above picture. This is a raw data, with this we can extract different types of report which is needed for the analysis.

Performance is the one of the key factor in project development, not only in case of software development, but also in all the industries.

This performance meter can be clustered and make to work with many number servers with the same test cases.

With the report produced from tool can make throughput of the given scenario.
This project is OS independent. We focus to make it as Web server independent one.

7.2 CONCLUSION

The above report details the work done regarding the Performance test Utility, “PERFORMANCE METER”. The various features supported by this feature have been documented. In spite of various solutions available for this feature in the market, this utility will be considered vital to the organization because of its customize option will be given to the project and the configuration can be done to distribute the performance meter to cluster and measure it with heavy load.

7.3 FUTURE ENHANCEMENTS

1. Support to take many different types of reports for the test cases.
2. Support for creating the graphical view for the test scenarios.
3. Support to make this work with clustered form (execute test cases independently).
4. Support for JAVA and other functionalities.
5. Support for many databases will be provided.
6. Support this application to various web servers.
7. Support to import and Export the test scenarios.

APPENDIX

Glossary

HTTP
DB
DAO
HOST
IP Address
PORT

This section provides the description or definitions of the terms used in Performance Meter.

HTTP

The **Hypertext Transfer Protocol (HTTP)** is a networking protocol for distributed, collaborative, hypermedia information systems. HTTP functions as a request-response protocol in the client-server computing model.

In HTTP, a web browser, for example, acts as a *client*, while an application running on a computer hosting a web site functions as a *server*. The client submits an HTTP *request* message to the server. The server, which stores content, or provides *resources*, such as HTML files, or performs other functions on behalf of the client, returns a response message to the client. A response contains completion status information about the request and may contain any content requested by the client in its message body.

DB (Database)

A **database** is a system intended to organize, store, and retrieve large amounts of data easily. It consists of an organized collection of data for one or more uses, typically in digital form. One way of classifying databases involves the type of their contents, for example: bibliographic, document-text, statistical. Digital databases are managed using database management systems, which store database contents, allowing data creation and maintenance, and search and other access.

DAO (Data Access Object)

A **data access object** (DAO) is an object that provides an abstract interface to some type of database or persistence mechanism, providing some specific operations without exposing details of the database. It provides a mapping from application calls to the persistence layer. This isolation separates the concerns of what data accesses the application needs, in terms of domain-specific objects and data types (the public interface of the DAO), and how these needs can be satisfied with a specific DBMS, database schema, etc.

Host

Host is a computer connected to a computer network. A network host may offer information resources, services, and applications to users or other nodes on the network. A network host is a network node that is assigned a network layer host address.

IP

The expansion of IP Address is Internet Protocol Address. A unique IP Address is Provided for each workstation, switches, printers, and other devices present in the Network for identification and routing of information.

Port

A **port** is an application-specific or process-specific software construct serving as a communications endpoint. It is used by Transport Layer protocols of the Internet Protocol Suite, such as Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). A specific port is identified by its number, commonly known as the **port number**, the IP address with which it is associated, and the protocol used for communication

REFERENCES

1. http://msdn.microsoft.com/en-us/library/bb924356.aspx - Scope of Performance Measure
2. http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html - Http request modals.
3. http://www.jdbc-tutorial.com/ - Database connectivity reference.
4. http://www.oracle.com/technetwork/java/index-jsp-142903.html - Java reference