



# YouTestMe

Resilience Testing

---

## Table of Contents

1	Introduction.....	3
2	Prerequisites.....	4
3	Testing Tools.....	5
3.1	Performance Testing .....	5
3.1.1	Testing Goal .....	5
3.1.2	Testing Scenarios.....	5
3.1.3	Testing Procedure.....	6
3.1.3.1	Setting Up Testing Environment.....	6
3.1.3.1.1	Loading Users .....	6
3.1.3.1.2	Creating Testing Session.....	6
3.1.3.1.3	Reconfiguring Environment for New Test .....	7
3.1.3.2	Running Test.....	7
3.1.3.2.1	Configuring testing target .....	8
3.1.3.2.2	Setting number of users .....	8
3.1.3.3	Monitoring.....	8
3.1.3.3.1	System monitoring .....	8
3.1.3.3.2	Log Monitoring .....	9
3.1.4	Testing Results.....	10
3.2	Java Resilience Testing .....	11
3.2.1	Testing Procedure.....	11
3.2.1.1	Recommended Test Scenario:.....	11
3.3	DB Analyzer .....	12
3.3.1	Testing Goal .....	12
3.3.2	Testing Procedure.....	12

## 1 Introduction

Software resilience testing is a software testing method that focuses on ensuring that applications will perform well in real-life or chaotic conditions.

In other words, it tests an application's resiliency or ability to withstand stressful or challenging factors.

In this particular case, the resilience testing will include the following tools and testing scenarios:

1. DB Analyzer - a tool for analyzing database servers, database structures, and their data
2. Java resilience testing - simple Java standalone application that generates a heavy load on the database by performing a resource-consuming operation such as copying large tables or updating its column with the new set of values
3. JMeter - allows you to simulate the test-taking process in a multi-user environment. The option "Retake test" can be convenient because it enables a virtual user to take the test over a relatively long period continuously.

## 2 Prerequisites

Database performance testing is impossible to do without a relatively large amount of data. For this purpose, PostgreSQL schema “ytm1\_resilience\_testing.dmp” was uploaded to the FTP server, and it should be imported into a target database for testing purposes.

/resilience_testing/				
Name	Size	Changed	Rights	Owner
..		10/27/2019 11:28:11 PM	rwxr-xr-x	5284759
dbanalyzer 1		10/28/2019 3:59:09 PM	rwx---r-x	5284759
java_resilience_testing 2		10/28/2019 4:05:49 PM	rwx---r-x	5284759
jmeter 3		10/28/2019 3:59:18 PM	rwx---r-x	5284759
postgresql_testing_database 4		10/27/2019 11:56:58 PM	rwx---r-x	5284759

The PostgreSQL 10 schema restore process consists of several commands:

1. Create a target database if it doesn't exist:

```
$ psql -h [DATABASE_IP] -U postgres -c 'create database testdb';
```

2. Create database role ytm1 if it doesn't exist:

```
$ psql -h [DATABASE_IP] -U postgres -c "create role ytm1 with password '2ytm1';"
```

3. Restore downloaded database schema in the testdb database:

```
$ pg_restore -v -h [DATABASE_IP] -U postgres -d testdb
ytm1_resilience_testing.dmp
```

4. Verify the status of the previous operation by connecting to the restored schema ytm1 and executing the following commands to retrieve the number of application users:

```
$ psql -h [DATABASE_IP] -U ytm1 testdb
testdb=# select count(*) from users;
```

5. Optionally, grant login privilege to role ytm1:

```
alter user ytm1 with login;
```

## 3 Testing Tools

### 3.1 Performance Testing

#### 3.1.1 Testing Goal

Performance testing shows how much load system can take without malfunctions. It covers usual usage scenarios that could lead the system to a point when the system is not manageable.

#### 3.1.2 Testing Scenarios

YouTestMe GetCertified is a complex system that may contain a reverse proxy, proctoring server, multiple application servers, and multiple database servers.

The performance of the system can be limited by many factors as a consequence of that.

Performance testing must be done in many different configurations.

That way, we can find the limits of different system parts and recognize potential bottlenecks.

Some of the scenarios that are tested include:

- Testing on different hardware resources
- Testing with/without Apache reverse proxy/load balancer
- Testing with/without SSL
- Testing on public and private network
- Testing on different physical hardware

JMeter is used as a platform for running tests.

JMeter job is to simulate a high number of users connected to the application and can perform tasks that users usually do.

Some of the JMeter testing scenarios are:

- Log in, log out
- Log in, start a test, finish a test, log out
- Log in, start a test, answer to N questions, log out

The test is considered successful if all users have passed all stages.

Another thing to consider is the number of JMeter stations.

Due to simulating high load on instances, the number of users can go over JMeter capabilities.

Multiple JMeter instances are used in testing with a high number of users on more advanced tests.

### 3.1.3 Testing Procedure

Before testing is started, we have to create an appropriate testing environment.

That includes deploying all servers needed for test scenarios, deploying the application version that is being tested, and configuring the database.

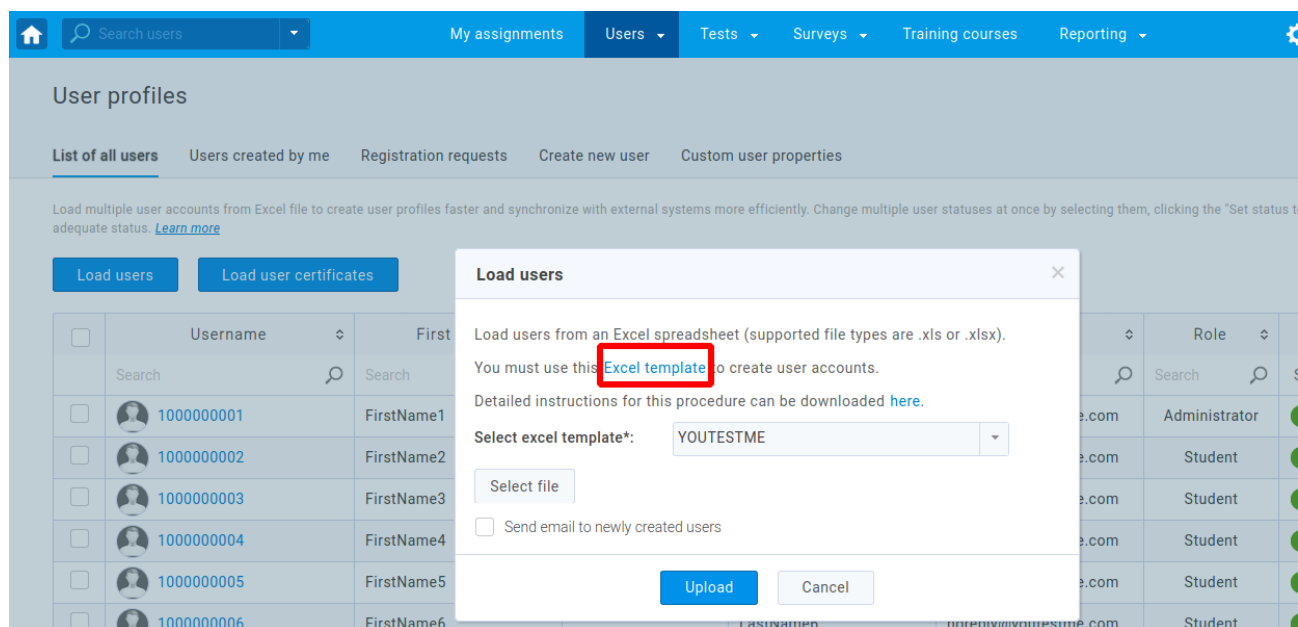
#### 3.1.3.1 Setting Up Testing Environment

##### 3.1.3.1.1 Loading Users

The simplest way to load the desired number of users is by using a prepared excel sheet.

The sheet can be downloaded from the application by going to Users -> Users profiles -> Load users.

The template is linked in the popup window.



After that, it is easy to create any number of users.

Usernames should be configured as numbers, so JMeter could iterate through users, and it is also easier to follow users progress.

Finally, load users by uploading Excel sheets with users using the same process.

##### 3.1.3.1.2 Creating Testing Session

One of the tests used is “Test\_40SNC\_Question\_UnlimitedTime\_Video” which should be loaded into the application.

In the tab Testing sessions, we create a new session.

After that, assign the desired number of users in the candidate tab.

Users cannot have previous attempts, as tests won't be found by the JMeter script.

### 3.1.3.1.3 Reconfiguring Environment for New Test

After running tests with JMeter, we want to reuse loaded users.

Resetting attempts could be done through the application, but it is more convenient to reset attempts directly through the database.

It can be done by executing the “reset\_session.sql” script on the application schema.

```
$psql -h DB_IP_address -U Schema_name -d DB_name -f reset_session.sql
```

After executing the script, all users are can take the test again, and the JMeter test could run again.

### 3.1.3.2 Running Test

JMeter test is started on the remote machine in the same or public network.

Before running the test, we have to set some parameters on the JMeter test file.

Configuring tests is done using JMeter GU or by initializing tests with parameters, but tests are started only using CLI.

After starting JMeter, go to File -> Open and find one of the testing scripts (e.g. Test\_40SNC\_Question\_UnlimitedTime\_Video.jmx).

In the left panel find **User Defined Variables**.

It will open a table similar to the image below.

User Defined Variables

Name: User Defined Variables

Comments:

User Defined Variables		
Name:	Value	Description
protocol	http	
ip	test.youtestme.com	
instance	ytm1	
port	8080	
startingUserId	1000000001	
password	Student12#	
thread	400	
rampup	600	
minWaitTimeBeforeSingIn	10000	
maxWaitTimeBeforeSingIn	20000	
minWaitTimeGoToMyAssignmentsPage	2000	
maxWaitTimeGoToMyAssignmentsPage	4000	
minWaitTimeToClickStartButton	4000	
maxWaitTimeToClickStartButton	8000	
minWaitTimeToReadInstructions	10000	
maxWaitTimeToReadInstructions	15000	
minWaitTimeToStartTest	3000	
maxWaitTimeToStartTest	5000	
minWaitTimeToSelectAnswer	10000	
maxWaitTimeToSelectAnswer	30000	
minWaitTimeToClickNextButton	5000	
maxWaitTimeToClickNextButton	10000	
minWaitTimeToClickFinishButton	5000	
maxWaitTimeToClickFinishButton	10000	
minWaitTimeToPreviewPerosnalReport	20000	
maxWaitTimeToPreviewPerosnalReport	30000	

### 3.1.3.2.1 Configuring testing target

Testing target parameters are

- **Ip** – URL or IP address of the server with the application. In case that we want to connect directly to the Tomcat server, we can use the IP address of the server or create a host link in the file /etc/hosts
- **port** – access port of the application. If the application is accessed directly, it is a Tomcat HTTP port. Port 80 or 443 is used when the application is hosted behind Apache reverse proxy
- **instance** – context name of the application in the case of the image, JMeter will access the application using http://test.youtestme.com:8080/ytm1

### 3.1.3.2.2 Setting number of users

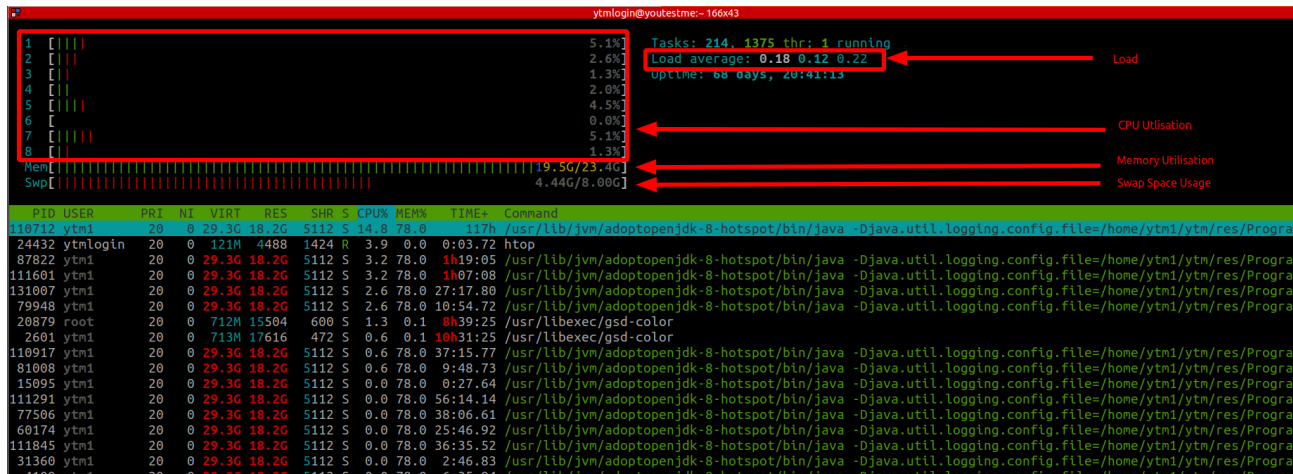
- **startingUserId** – First user that will be started on the JMeter testing.
- **Threads** – number of time JMeter will do everything in the script. Every thread starts with a new user.

### 3.1.3.3 Monitoring

#### 3.1.3.3.1 System monitoring

System monitoring is done using the program 'htop'. It is a simple CLI program that presents all necessary information about the current state of the server.





Important parameters:

- *Load* – shows the average load on the system in the last 1/5/15 minutes. The value shouldn't exceed the number of vCPUs assigned to the server
- *CPU utilisation* – shows percentage used of total core capacity. Values shouldn't get in saturation
- *Memory utilisation* – shows the percentage of memory used on the system. High memory consumption can force the system to use swap space
- *Swap space usage* – shows the percentage of swap space occupied. Swap space is used to increase server memory capacity. It uses space on local storage and is much slower than actual server memory (RAM). The system shouldn't start using Swap storage.

### 3.1.3.3.2 Log Monitoring

Log files can alert on apparent system failures due to bad configuration.

Files that are monitored during testing are:

- Application and Tomcat log file (catalina.out)
- PostgreSQL log file
- Apache error file

### 3.1.4 Testing Results

	Scenario #1	Scenario #2	Scenario #3
<b>Steps</b>			
Test step #1	Log in	Log in	Log in
Test step #2	Start test	Start test	Start test
Test step #3	Answer 0 questions	Answer 0 questions	Answer 5 questions
Test step #4	Submit test	Submit test	Submit test
Test step #5	Log out	Log out	Log out
<b>Configuration</b>			
Proxy Server	ON	OFF	OFF
Load Balancer	ON	OFF	OFF
SSL	ON	OFF	OFF
Network	Public	Private	Public
JMeter mode	Single	Distributed (4 x 250)	Distributed (4 x 250)
Virtual Cores	4	4	4
Virtual RAM (GB)	16	32	32
Physical Hardware	8 CPUs x Intel(R) Xeon(R) CPU E5-1660 v3 @ 3.00GHz	8 CPUs x Intel(R) Xeon(R) CPU E5-1660 v3 @ 3.00GHz	8 CPUs x Intel(R) Xeon(R) CPU D-1540 @ 2.00GHz -64 GB RAM, HDD Storage
<b>Outcome</b>			
Number of Test Attempt	4	2	2
<b>Number of Concurrent Users</b>	<b>800</b>	<b>1000</b>	<b>1000</b>
Number of Logins	800/800	1000/1000	1000/1000
Number of Users Started Test	791/800	1000/1000	1000/1000
Number of Finished Tests	536/800	1000/1000	1000/1000

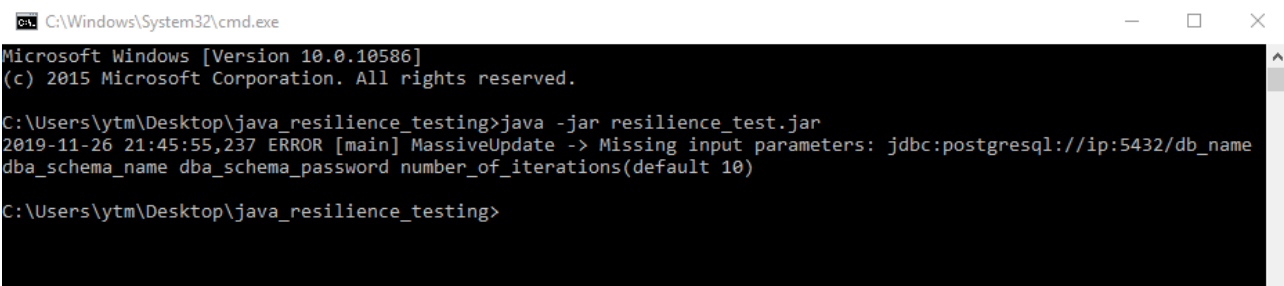
## 3.2 Java Resilience Testing

### 3.2.1 Testing Procedure

To start the Java resilience test, download content of the “java\_resilience\_testing” directory and:

1. Execute the following command (JDK 8 required):

```
$ java -jar resilience.jar
```



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\ytm\Desktop\java_resilience_testing>java -jar resilience_test.jar
2019-11-26 21:45:55,237 ERROR [main] MassiveUpdate -> Missing input parameters: jdbc:postgresql://ip:5432/db_name
dba_schema_name dba_schema_password number_of_iterations(default 10)

C:\Users\ytm\Desktop\java_resilience_testing>
```

2. Enter the required parameters, similar to the example below:

```
$ java -jar dba.jar jdbc:postgresql://IP:5432/testdb ytm1 2ytm1 100
```

3. After the proper database parameters were specified and the test is started, ten threads will be created and run concurrently. Each of them will perform the following operation in each iteration:
  - a. create a copy of the correspondent table with the highest number of rows
  - b. delete it

#### 3.2.1.1 Recommended Test Scenario:

1. Take a database server snapshot before resilience testing
2. Start the test by specifying a relatively high number of test iteration (1000, or even more)
3. If the test was completed successfully, compare the system parameters before and after the test was taken and check if there is no free disk space or if the RAM reached its limit.

This type of test can expose some database misconfiguration or prove that an insufficient amount of hardware resources (physical or virtual) were allocated to a database server.

## 3.3 DB Analyzer

### 3.3.1 Testing Goal

DB Analyzer can be used as a useful tool for database performance testing.

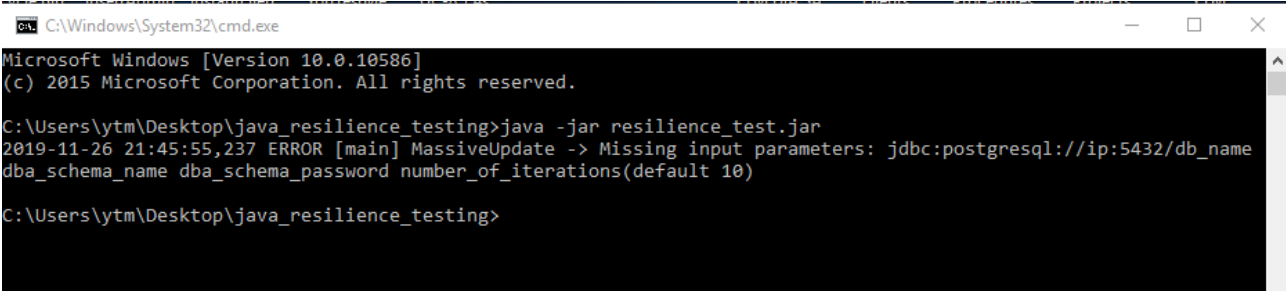
Critical errors in database design and nonoptimized views will cause a significant increase in total runtime, so DBA can help you to identify them.

### 3.3.2 Testing Procedure

To start DB Analyzer from the command line, please download the content of “DBAnalyzer” directory, and:

1. Execute the following command (JDK 8 required):

```
$ java -jar dba.jar
```



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

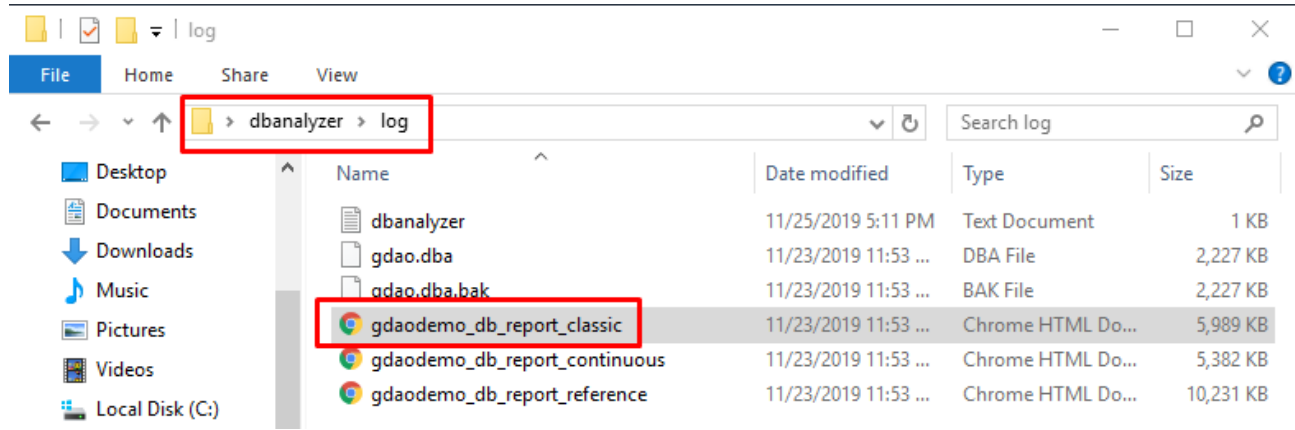
C:\Users\ytm\Desktop\java_resilience_testing>java -jar resilience_test.jar
2019-11-26 21:45:55,237 ERROR [main] MassiveUpdate -> Missing input parameters: jdbc:postgresql://ip:5432/db_name
dba_schema_name dba_schema_password number_of_iterations(default 10)

C:\Users\ytm\Desktop\java_resilience_testing>
```

2. Enter the required parameters, similar to the example below:

```
$ java -jar dba.jar jdbc:postgresql://IP:5432/testdb ytm1 2ytm1 100
```

3. Wait for the program to complete its analysis
4. Check the produced log files. They will contain many useful information, such as a list of database objects and their size, redundant objects, detailed table information, etc.



Note: The additional file “gdao.cfg” specifies the advanced functions of DBA, which are not covered in this document since the configuration is already setup to support in-depth database analysis.