# **Lite Weight Reporter**

# **Overview**

How many times have we run SQL queries on RDBS using rich SQL editors to extract data, copy it to excel sheet to plot simple trend charts? Why do we not use any of the existing BI or reporting tools?

The overhead of using heavy weight BI tools is little too much for a simple report/charting requirement.

Firstly, you need to have a licensed version, trial versions are good but they don’t offer continuity.

Secondly, there is a learning curve associated in building reports no lets you transform just the SQL queries to chart directly, you need to know the product in building even a simple report.

Lite weight reporter addresses both these limitations. It’s an open source, based reporting solution that lets you map SQL queries to beautiful reports and charts.

Light weight reporter, LWR, is a simple web based reporting building and publishing application. LWR transforms bunch SQL queries to charts and displays in a report. Each report has one or more elements arranged in rows and columns. Each element has title, SQL Query, database connection pointing to database from where data is retrieved and type of graph/table renderer.

**Charting Libraries**

The application uses google charts (default) or jfree charts to build the charting components. The charting library can be changed between google and jfree via a user level preference setting so that users have their own charting library preference. The chart element supported by both these charting libraries are

1. Bar Chart

2. Bar Stack Chart

3. Column Chart

4. Column Stack Chart

5. Line Chart

6. Pie Chart

7. Tabular Chart

User has to define database connection by providing the username, password, driver and connection URL. This connection is later used to create report element. We can define multiple database connection pointing to different databases and use them in different elements of a report thus retrieving data from different databases. Each user has an option of making one of the connection as default so that same report can retrieve data from different database based on user's default database. For example production users for whom the default database is marked as the production database can see data coming from production database while the non-production users see the same report but data coming from non-production database.

When the report is rendered in browser, each element makes an Ajax call to retrieving data from database and rendering them into a chart. What gets loaded when user clicks on a report is the report template while the actual elements are rendered in parallel hence user doesn’t have to wait un till the entire report is rendered.

# **Features**

## **User Experience**

The user interface of the application is built using HTML5 and bootstrap for styling. No java applets are used enhancing user experience with faster performance.

Report building can be done latest version of any browser and hence doesn’t require any rich java client to be installed on the laptop/desktop.

Report elements are loaded in parallel using AJAX and hence user need not wait for entire report to be rendered before see data/charts for simpler and smaller reporting elements.

For example, report has two element, one a simple table with 5 rows and another line chart component which loads last one month data. Since the report is rendering using AJAX calls, the simple table gets displayed immediately when user clicks on the report while the expensive line chart components takes time to load.

## **Real Time Reporting Using Auto Refresh**

Most of the time we want to see the latest data coming into the database at real time in reports without having to keep refreshing the report every few seconds/minutes.

LWR has auto refresh functionality available at individual element of the report. By clicking on the desired element of the report, one can make that element retrieve and render data in real time. The auto refresh is every 5 minutes by default. The refresh interval is configurable and can go down up to 1 second.

The idea behind providing the refresh at element level and also on demand is to prevent resource over utilization. User might want to see only one component’s/element’s data in real time and in such cases refresh entire report can be an over kill.

## **Distributed & Scalable**

LWR backend logic is implemented using RESTful web service and front end of the application uses HTML, Java Server Pages and Java Script with AJAX. Both these components be hosted on different servers independently.

The application uses connection pool hence administrator can restrict the number of connections that the application can make to database. If the database connections are exhausted, the corresponding report query is queued. The connection pool is maintained per database connection defined.

## **Data Export**

The report can be exported in to two formats, CSV and static HTML.

## **Report Scheduling**

Reports can also be scheduled and delivered over email or available in user’s personal folders in either of the CSV or HTML formats.

# **Getting Started**

## **Prerequisite**

* Tomcat Application Server v8.0 and above
* Required JDBC Driver(s) to connect to database(s)

## **Installation**

Copy the lwr.war file downloaded from sourceforge.net and place it under CATALINA\_HOME/webapps/ directory. Lite Weight Reporter uses JDBC to connect to database(s). The JDBC drivers for Postgres and MySQL are already bundled into the war file. To connect to any other database vendor, copy the required vendor specific JDBC jar file to CATALINA\_HOME/webapps/lwr/WEB-INF/lib folder and restart the tomcat server

## **Building First Report**

1. Access the application using below URL,   
   **http://<hostname>:<port>/lwr**  
   Where port -> port on which tomcat is listening
2. Once you open this URL in your browsers you would be redirected to login page that asks you for username and password. The default username is **"admin"** and default password is **"admin"**. It is highly recommended to change the password for "admin" user by navigating to Administration->User Management page. You can create more users.
3. Create a database connection by going to **Administrator -> Connection Management**. Connection is created by providing

* Username
* Password
* JDBC Driver Class
* JDBC Driver URL

Once these properties are keyed in, performn test connection to check if the connectivity is established using above mentioned properties. If test connection is successful, save the connection details.  
The application lets you define one or more such connections pointing to different databases and within a report we can pick data from different connections.

1. Building a report by going to **File -> New Report** this involves multiple elements arranged in rows and columns. For each of the element you need to provide

* Title
* SQL Query
* Chart Type
* Database Connection

A single report having multiple elements can retrive data from one or more databases. Support Chart Types are

* Pie Chart
* Line Chart
* Bar Chart
* Bar Chart Stacked
* Column Chart
* Column Chart Stacked
* Table Chart

All these charts are google charts and you need to be connected to internet for it to render the HTML