Margin notes version!!

Stairwalker Manual

We could write something here

Dennis Muller
Jochem Elsinga

October 17, 2014
Universiteit Twente $\boxed{\mathcal{UT}}$

Contents

1 Introduction

This manual will explain how the Stairwalker Program of the Database section of the University of Twente works. This manual will start by explaining on how to install all required components in order to have the basic running. This includes how to install all the required extensions for your database but also for other applications that are required to run this system. The second section will explain how to add all databases and different kind of datatypes to Geoserver, which is required to show in order to make the Stairwalker program run. Not only is this section going to explain how to show the data but it will give the option to customise this view and have it display data on all possibly ways. And the third section will explain how to make adjustments to the existing program and what future possibilities the system has. For example on how to add a new dimension to the aggregation tool or how to use a different aggregation type (median for example). The last section covers further development on the Stairwalker program incase something that hasn't been made yet needs to be added. This manual is made for a Postgres database other database programs are also possible, however this manual will not cover how to instal certain extensions on those programs.

2 Set Up

In this section a detailed description will be given about setting up all the required peripheral programs to use stairwalker.

2.1 Database Setup

At the moment the Stairwalker program only works with the PostgreSQL[1] and MonetDB[2] database management systems. This manual will only concern itself with PostgreSQL (shortened to Postgres). Any further reference to a database implies a Postgres database.

In this section a walk through explaining the steps needed to setup PostgreSQL along with PostGIS[3] and a custom made database extension for Stairwalker on an Ubuntu Linux kernel.

2.1.1 PostgreSQL Database Manager Setup

Installing Postgres on Linux should be straight forward. In the terminal search for PostgreSQL and then choose which version of PostgreSQL should be installed. Below the commands used to search and install version 9.1 of Postgres are shown.

1. `aptitude search postgresql`
2. `apt-get install postgresql-9.1`

For further information or if there are difficulties with the installation more details can be found on the installation page of the PostgreSQL website.

2.1.2 PostgreSQL Configuration

Once Postgres is installed on Linux two alterations will need to be made to the configuration files so that the database can be accessed from outside and by other users.

First off in the `Postgresql.conf` file the listen_addresses need to be changed from localhost to all. This can be done as follows (assuming version 9.1 of Postgres).

1. `cd /etc/postgresql/9.1/main`
2. `vi postgresql.conf`
3. *change* listen_addresses = 'localhost' → listen_addresses = '*'
4. *save and close*

[1] PostgreSQL: `http://www.postgresql.org`
[2] MonteDB: `http://www.monetdb.org`
[3] PostGIS: `http://www.postgis.net`

Secondly in the file `pg_hba.conf` a line needs to be added to allow other users in Linux to access postgres. This can be done as follows:

5. `vi pg_hba.conf`
6. *Add* host all all 0.0.0.0/0 password
7. *save and close*
8. `/etc/init.d/postgresql restart`

It should now be possible to create a database user and database in Postgres. More information about how this is done can be found on the Postgres manuals webpage.

2.1.3 PostGIS Configuration in PostgreSQL

The next step is to extend Postgres with PostGIS database expansion. This again should be straight forward, first search for PostGIS and then choice the version that goes with Postgres to install. Note in order to do the install root privileges are required. Below the commands to search and install PostGIS for Postgres-9.1 are shown.

1. `aptitude search postgis`
2. `apt-get install postgresql-9.1-postgis`

More information about installing PostGIS can be found on the PostGIS website.

Once PostGIS is installed some extra configuration is required. This needs to be done as postgres user. The commands are as follows.

Why do we do this again? What does it do?

1. `createlang plpgsql geonames`
2. `psql -f 'find/usr/share/postgresql/ -name postgis.sql -print' -d geonames`
3. `psql -f 'find/usr/share/postgresql/ -name spatial_ref_sys.sql -print' -d geonames`
4. `psql -f 'find/usr/share/postgresql/ -name postgis_comments.sql -print' -d geonames`

2.1.4 Serverside Stairwalker Extension in PostgreSQL

Finally there is C extension which allows processing of the SQL queries for the pre-aggregated database to happen on the server side instead of in the GeoServer application. This extension has to be installed on the Postgres database. The extension can be found in the directory `neogeo/pre-aggregate/src/db-extensions/postgres/pa_grid`.

The extension can be installed on Linux using the following commands.

1. *go to the pa_grid directory in the terminal*
2. `make`
   this creates the dynamic library
3. `make install`
   this installs the library in the postgres installation
4. `make sql`
   declare the module in the database wanted

Note the extension has to be installed specifically for the database which will be used for pre-aggregation. In the makefile (also in the `pa_grid` folder) there is a `DATABASE` macro which should be set to the desired database.

Installing the extension also requires root access. Why was this again?

2.2 Pre-Aggregate Database Table

2.2.1 Description of Process

Stairwalker makes use of the Pre-Aggregate functions. These functions create a granularity in the dataset and calculate blocks at the lowest granularity for certain dimensions. Then subsequent blocks of higher granularity are built up from lower blocks. This creates a new dataset which as summarized the original dataset in terms of blocks which represent some significant data for every level of granularity and dimension.

Concretely take the dataset of tweets sent within the netherlands. A pre-aggregation of this data could be in the following form. As significant data we consider the number of tweets in the x- and y-coordinate dimensions. We then set a highest granularity (zoom). The pre-aggregate algorithm then creates blocks bounded by the x- and y-coordinates at the highest granularity and for all those blocks calculates the number of tweets within those coordinates. Then each subsequent layer is built up from the previous layer. The final result is a new dataset which contains all blocks with the number of tweets in a coordinate block at each level of granularity.

2.2.2 PreAggregate Tool

To do the Pre-Aggregation step a tool has been develped which is explained here.
The tool can be found in the directory `neogeo/pre-aggregate-tools/`. To generate the binary tools from source the Appassembler plugin of Maven is used. Run the following command to generate the tool.

```
mvn package appassembler:assemble
```

After successful completion of this command a new directory `appassembler` will have been created in the `target` directory containing a `repo` and a `bin` directory. The `bin` directory contains the actual binaries of the tool (in both Linux/Unix and Windows version) and the `repo` directory contains the tool dependencies. The tool should now be ready for use.

The tool is used to create a PreAggregate index for a table with n dimensions and a measure/aggregate column. It uses with the following commands:

```
usage: create-pa-index
 -axistosplit <axis index>     index of axis to split
 -chunksize <size>             maximum chunk size after split
 -config <file>                PreAggregate XML config file
 -d,--database <dbname>        name of database
 -dbtype <postgresql|monetdb>  type of database
 -h,--host <host>              database host name or ip address
 -help                         prints this help message
 -p,--port <port>              port number of the database
 -password <password>          database password
 -s,--schema <schema>          schema name in the database
 -u,--user <user>              database username
 -v,--verbose                  Enable verbose output logging
```

The tool is dependent on the `PreAggregate.XML` config file which is used to define the PreAggregate index by specifying the column to aggregate, the type of aggregate that should done and the dimensions to include. In the `neogeo/pre-aggregate-tools/` directory a sample configuration file is included.

How does this tool work nominal axis? More preprocessing may be required when first parsing words? Example london_words or gender_words

see Section 4.1 for more detail about preaggregation not using the special help tool.

Apart from creating a new PreAggregate table `<original-table-name>_pa` pre-aggregation also creates/updates two other tables, these are `pre_aggregate` and `pre_aggregate_axis`. These two tables are support tables for the aggregation. They contain information about which tables have been aggregated and which axis have been used in pre-aggregation.

2.2.3 Current Status Support of Datatypes

1. Aggregate Axis

2. Metric Axis

3. Nominal Axis

2.2.4 Current Status Support of Aggregation Types

At the moment you can aggregate your data in a total of 4 options and all 4 options have a different outcome.

The options that you can use at the moment are the following:

1. ALL: ???

2. Count: returns the total amount of data in an aggregate box. Returns a number with the total amount of that tile.

3. Sum: returns the total value of all data added up to eachother. For instance when you do a sum on the tweet length you'll get the total amount of characters tweeted in a tile.

4. Min: returns the lowest value of all data that is aggregated. With the example of tweet length, it will returns the value of the lowest length tweet in that tile.

5. Max: returns the highest value of all data that is aggregated. With the example of tweet length, it will returns the value of the highest length tweet in that tile.

It is important to chose the right type in order to get a good representation of your data. If you want to show the total amount in a tile, you should chose count as this will returns the total number of tweets in a tile. Whereas if you want only the highest value of your data (for example the highest building in an area) then it is important to use max. It is also possible to add other datatypes than the onces mentioned here. To do so check the section about development, here you will find more information about how to add a different aggregation type.

2.3 Tomcat Server Setup

To visually display the aggregated dataset a third party application is used. This application is called GeoServer, GeoServer is an open source server which can be used to display geospatial data. In order to run GeoServer a web server is needed.

During development the Apache Tomcat[4] was used for this purpose. The use of Tomcat is not required for stairwalker alternative options may also be used as long as it is possible to deploy WAR files. Because Tomcat was used during development a description of how to obtain Tomcat is given here. Furthermore

[4] `http://tomcat.apache.org/`

whenever web hosting is mentioned in this manual it will be assumed Tomcat is being used.

Information about setting up a Tomcat server can be found on the Apache Tomcat website.

2.4 Geoserver Deployment on Tomcat

2.4.1 Obtaining GeoServer and Aggregate Extension

The GeoServer[5] is used to visually display the aggregated data. To use the aggregated dataset an extension has be written for GeoServer which needs to be included in the web application. This is done by including the JAR files of the extension in the WEB-INF of the GeoServer WAR file. A step by step guild is given below in Section 2.4.2. However before following these instructions the necessary JAR and WAR files will need to be obtained.

The GeoServer WAR file can be downloaded from the GeoServer website. The custom Java extension files should be built using the source code. The extension consists of the `pre-aggregate` and `geoserver-ext` projects in the `neogeo` project. Note that first the JAR file of `pre-aggregate` should be created as it is a dependency of `geoserver-ext`. The JARs can be built using the command line or in a IDE such as Eclipse or Netbeans, using the `clean and build` command on a project in Netbeans should build `<project-name>-0.0.1-SNAPSHOT.jar` which can be found in the `target` directory of the respective project.

2.4.2 Installing Extension

These next instructions assume the geoserver extention files are located in the directory `/data/upload/` and that the `geoserver.war` file is located in the directory `/data/tmp/tmp_war`. If this is not the case the file paths in the instructions below should be changed accordingly.

Unpack the WAR file

1. `jar -xvf geoserver.war`

Copy the JAR files of the extension into `WEB-INF/lib` directory of the GeoServer unpacked WAR file

2. `cp /data/upload/pre-aggregate-0.0.1-SNAPSHOT.jar /data/tmp/tmp_war /WEB-INF/lib`
3. `cp /data/upload/geoserver-ext-0.0.1-SNAPSHOT.jar /data/tmp/tmp_war /WEB-INF/lib`

Recreate the GeoServer WAR file

4. `jar -cvf geoserver.war META-INF/ WEB-INF/ index.html data/`

[5] GeoServer: `http://www.geoserver.org`

The GeoServer WAR with the stairwalker extension should now be ready for deployment.

2.4.3 Deploying GeoServer

After the GeoServer WAR file has been repacked with the aggregate extension included it is ready to be deployed in a web server. Section 2.3 describes how to set up such a Tomcat web server. Once the server is running it can be accessed locally with `http://localhost:8080/` assuming default installation configuration were used, otherwise the port number might be different. In the case that the Tomcat server is installed on a difference machine then the web server can be accessed by replacing `localhost` with the IP address of that machine.

From the Tomcat homepage it should be possible to access the Tomcat manager webapp. With the default setup it should be possible to login with the following credentials:

username: manager

password: tcmanager

In the Tomcat manager webapp under the `deploy` section it is possible to upload a WAR file to be deployed. Select the repacked WAR file from Section 2.4.2 and deploy the application. Once the application is deployed it will be displayed in the `application` section of the Tomcat manager webapp. From there it is possible to follow the path given for the GeoServer application or, if the default configuration was used to go to `http://<Tomcat-IPaddress>:8080/geoserver/web/`.

3 Deployment

This section will give a detailed description of how to import an aggregated database table into GeoServer to get a visual representation of the dataset. First instructions will be given on how to link the database table to GeoServer. Next creating styles and layers for data representation will be discussed. The final sub section will discuss how to view the data using GeoServer. For this section it is assumed that all the initial preparation discussed in Section 2 has been completed.
newline

GeoServer should already be deployed on a web server (see Section 2.4.3), and can then be accessed with `http://<Tomcat-IPaddress>:8080/geoserver/web/`. It is required to login in to the GeoServer administration interface. When using the default setup of GeoServer the login credentials are:

username: admin
password: geoserver

3.1 Add Source

Once logged in to the web administration interface it is possible to add a new data store to GeoServer. Instructions of how to add a new `NeoGeo Aggregate` vector data source which contains the aggregate index created in Section 2.2.2 to the stores in GeoServer.

1. Navigate to `Stores` by clicking on `Stores` link under the `Data` section in the navigator on the left hand side of the web administration interface homepage.

2. On the `Stores` page select the option `Add new Store` located at the top of the page. This leads to a page titled `New Store chooser`.

3. In the list of `Vector Data Sources` the option `NeoGeo Aggregate` should be present, choose this format for the data source.

If the option `NeoGeo Aggregate` is not available it means the GeoServer extension from Section 2.4.2 was not done correctly.

4. Clicking `NeoGeo Aggregate` will open a new page titled `New Vector Data Source` in which several fields have to be filled out, explanation of mandatory fields can be found in the list below on page 12.

5. For an express setup the fields which have already been filled can remain the same.

6. Once all the required fields are filled in click the `Save` button.

7. A new `NeoGeo Aggregate` source is now created and can be view and edited in `Sources`.

8. After saving GeoServer opens the page `New Layer` on which new layers can be created using the `Data Source`. How this is done is discussed in Section 3.3.

Here a list of all mandatory fields on the `New Vector Data Source` page with explanation.

- **Data Source Name** - An arbitrary name which will be assigned to the store.
- **Database type** - The type of underlying database, either PostgreSQL or MonetDB.
- **Hostname** - Hostname of the database server where the aggregation index is maintained.
- **Port** - Port number of the database.
- **Schema** - Name of the schema where the aggregation index is maintained.
- **database** - Name of the database where the aggregation index is maintained.
- **Username** - Username of the used database.
- **Password** - Password of the used database.
- **xSize, ySize, timeSize** - Specifies the dimensions of the grid which is created for every view of the map to calculate the aggregates per cell. The higher the number of cells the more detailed the information.

  Are the xSize, ySize, timeSize dimensions not already set in the source? What happens when timeSize > 1?

- **count, sum, minimum, maximum** - Select the boxes of the aggregates which will be used in the visualization. Note that these basic aggregates more aggregates such as mean can can be derived.
- **Enable server-side Stairwalker** - Selecting will cause the data source to rely on the use of the database plugins to use the Pre-Aggregate index. For performance reasons it is high recommended to use this option. See Section 2.1.4 for more details.

  The Enable server-side Stairwalker ooption is no longer avaible? Remove?

- **Enable query logging** - Selecting will turn on the logging of all Pre-Aggregate queries into a separate table called `pre_aggregate_logging`.

3.2 Setup Style

In GeoServer styles are used to render, or make available, geospatial data. Styles are used to visually represent the aggregation index which is represented in a layer. In GeoServer layers are written in Styled Layer Descriptor (SLD) which is a subset of XML. GeoServer comes setup with several different styles however to get the most out of the dataset it is best to develop a style specific to the layer which represents that data.

In this section only instructions on how to add new styles to GeoServer is given. For information on how to edit styles see Section 4.2 or the GeoServer user manual[6] which gives an in depth guide on developing styles.

1. Navigate to `Styles` by clicking on the `Styles` link under the `Data` section in the navigator on the left hand side of the web administration interface homepage.
2. On the `Styles` page select the option `Add a new style` located at the top of the page.
3. A new page titled `New Style` should open. There are now two possibilities, either a new style can be developed completely in the browser or a SLD file can be imported.
4. To import an already created SLD file scroll to the bottom of the page and press the `Choose File` button.
5. Select the style which should be imported and then press `Upload...` in the browser.
6. This fills in the `Name` field with the name of the file and the SLD editor with the content of the file.
7. It is possible to check the syntax of the SLD code by pressing the `Validate` button at the bottom of the page. At the top the page GeoServer will give feedback on the SLD code, either error messages or a no validation errors message.
8. Finally the style can be saved by pressing the `Submit` button at the bottom of the page.

We should maybe mention something about the workspace? What does `nurc` represent? It is even important to choose a workspace?

3.3 Adding Layer

Adding a layer can be done as follows:

Give some context about what a layer is

[6] `http://docs.geoserver.org/stable/en/user/styling/index.html#styling`

1. Navigate to `Layers` by clicking on the `Layers` link under the `Data` section in the navigator on the left hand side of the web administration interface homepage.
2. On the `Layers` page select the option `Add a new resource` located at the top of the page.
3. This leads to a new page where the `Store` which contains the layer needs be chosen from a drop-down list. If there are no `Store`s available make sure one was added, see Section 3.1
4. Choose the `Store` in which the aggregation index is stored.
5. Once a `Store` is selected a list of resources contained in the `Store` is given. These resources are the different aggregated indexes in the database which was linked to a `Store` in Section 3.1.
6. Select the pre-aggregate index which should be visualized in a layer by clicking the `Publish` link corresponding to the `Layer name` of the aggregate index.

At this point a layer has been selected to be published. This layer will be a visual representation of the data from the aggregation index created in Section 2.2.2. In order to make sure the correct geographical location is used in GeoServer and to give the layer a fitting style the following steps have to be taken in on the `Edit Layer` page.

This might be a good point to reference/show the running example.

7. In the `Data` tab the following sections and fields should be filled in.
   (a) In the `Basic Resource Info` there are some labeling fields. Standard the `Name` and `Title` are `<aggregation-index-tablename>` followed by `___myAggregate`. These can both changed to whatever is desired. However make sure the that the `Enabled` box is ticked in this section.
   (b) In the `Coordinate Reference Systems` section there are three fields.
      i. `Native SRS` should be `EPSG:4326`.
      ii. `Declared SRS` should be `ESPG:3857`. This coordinate system is used since it is what is usually used for tile based map representation.
      iii. `SRS handling` should be `Reproject native to declared`.
   (c) In the `Bounding Boxes` the coordinates corresponding to the data from the aggregation index is calculated for GeoServer. For `Native Bounding Box` click `Compute from data` and for `Lat/Lon Bounding Box` click `Compute from native bounds`.
   (d) All other sections in this tab are of little importance in a basic deployment.

8. Next in the `Publishing` tab a style can be added to the layer, the default style of a layer is `polygon`. In the section `WMS Settings` the field `Default Style` can be changed by selecting the desired style from the drop-down menu. For more about styles and creating styles see Section 3.2 and Section 4.2.

If the aggregation index does not contain a time dimension the setup of the layer is now complete and can be saved. However if the aggregation index does have a time dimension some additional adjustments need to be made which are described below.

9. Select the `Dimensions` tab.
   (a) Enable the the `Time` dimension.
   (b) As `Attribute` select `starttime`.
   (c) Do not set an `End Attribute`.
   (d) As `Presentation` select `Continuous interval`.
10. Save the layer.

Some concluding text

3.4 Viewing and Using Layer

This section shows how to view a newly created layer. A bit is said about extending the URL with &VIEWPARAMS to filter nominal axises. It also explains how a layer was used in the development/demo in a webpage. References to Client Side Development from section 4.

1. Navigate to `Layer Preview` by clicking `Layer Preview` link under the `Data` section in the navigator on the left hand side of the web administration interface homepage.
2. The `Layer Preview` page will have a list of all configured layers with can be previewed in various formats.
3. Locate the layer which should be shown and from the `All Formats` column choice any `WMS` format.
4. After selecting a format to view the layer a new page will open with a visual representation of the top most layer of dataset.

Errors is viewing the layer in GeoServer could be because, code error (fault in axis) or a style issue

4 Development

4.1 Code Development

4.2 Geoserver Visualization

This section is going to discuss most of the possibilities that GeoServer has to offer when it comes to the visualization of the data. The GeoServer manual has some information on this subject, which can be found on: `http://docs.geoserver.org/2.5.x/en/user/styling/sld-reference/index.html`.

The following sections will give some more information on what the differences are between the different options and some ways to implement the different options. This section will use some of the examples used in our demo to give some more insight what can be done and hopefully making it easier to realise what is wanted. Most information can be found on the website, below will be a discussion on what types are best used when, but also some more information on how to make them work properly.

4.2.1 Symbolizers

In SLD there are three different symbolizers, a linesymbolizer, a pointsymbolizer and a polygonsymbolizer. A pointsymbolizer is used when the data that has to be represented is best shown as points. It does exactly what it says, you'll get a map with points on it and each point will represent a data-object from your dataset. This symbolizer can be really handy in certain situations. For example when you want to show all locations were a rare species of a bird has been found it will show a map with all points were a bird has been reported.

A linesymbolizer is used when the data that has to be displayed is best shown in lines. This symbolizer is best used to display roads for example. It isn't a symbolizer that can be used to represent data very well, but it is more used for pre-defined data, think off rivers, roads etc.

A polygonsymbolizer is used when the data that you want to represent has to be displayed in two-dimensional objects. There are many possibilities in a polygonsymbolizer. It is possible to make a simple square but it also has the option to make circles or triangles. It is one of the most commonly used symbolizers. It has the most potentional since you can do a lot with a polygonsymbolizer. A good example where a polygonsymbolizer is used, is to display the amount of people living in cities. This can be done with a circle polygon and that the circle will get bigger when more people live in a city.

4.2.2 Filters

Filters are the most important function when it comes to making a custom style. A filter is basically the basis of a fancy layer. What a filter does is that it makes a ruling and if that ruling is met, the color, labeling etc will be done. In SLD it is possible to have an unlimited amount of filters so the possibilities are endless. The following filter expression can be used:

- <PropertyIsEqualTo>
- <PropertyIsNotEqualTo>
- <PropertyIsLessThan>
- <PropertyIsLessThanOrEqualTo>
- <PropertyIsGreaterThan>
- <PropertyIsGreaterThanOrEqualTo>

An example on how a single filter can be used is the following:

```
<ogc:Filter>
  <ogc:PropertyIsLessThan>
    <ogc:PropertyName>testvalue</ogc:PropertyName>
    <ogc:Literal>200</ogc:Literal>
  </ogc:PropertyIsLessThan>
</ogc:Filter>
```

This example will test if the `testvalue` is less than 200. If this is the case you can add what the filter should be doing. Below is the complete example that does something with this filters.

```
<Rule>
  <Name>SmallPop</Name>
  <Title>Less Than 100</Title>
  <ogc:Filter>
    <ogc:PropertyIsLessThan>
      <ogc:PropertyName>testvalue</ogc:PropertyName>
      <ogc:Literal>100</ogc:Literal>
    </ogc:PropertyIsLessThan>
  </ogc:Filter>
  <PolygonSymbolizer>
    <Fill>
      <CssParameter name="fill">#38FF19
      </CssParameter>
      <CssParameter name="fill-opacity">1.0
      </CssParameter>
    </Fill>
    <Stroke>
      <CssParameter name="stroke">#000000
      </CssParameter>
      <CssParameter name="stroke-width">1.0
      </CssParameter>
    </Stroke>
  </PolygonSymbolizer>
</Rule>
```

What this example does is that if the `testvalue` is below `100`, it will fill a polygon with the color: `#38FF19`. If this is not the case it will go to the next rule (if there is any otherwise it will just not do anything). The image shows a graph of the implementation we made for our data. The image has different kind of colors for the amount of data in a tile. If the amount is high the color will become more red and if there is little data the tile will be green. This is a good example on what can be done with filters.

4.2.3 Extra Options

GeoGerver SLD has a lot of options when it comes to customizing the data display that you've made. Below are some of the important features that are commonly used in GeoServer.

Halo: A halo gives a glow behind the current label. It should always be used in a textsymbolizer, since this is the only place you can add a halo. To use a halo its very simple, you do `<Halo> </Halo>` and in between it is possible to add `<Radius>` and `<Fill>`. For more information on how to use a `<Fill>` look in the GeoServer SLD cookbook.

Anchorpoint: An anchor point is a really handy tool to place your label on every place possible. It is used as shown below and important to notice is that you can set where the anchor point is (for example above the point) and you can displace it afterwards based on this anchor point, for instance make it go all the way to left (negative X placement) or all the way to the right (positive X placement)

```
<PointPlacement>
  <AnchorPoint>
    <AnchorPointX>0.5</AnchorPointX>
    <AnchorPointY>0.0</AnchorPointY>
  </AnchorPoint>
  <Displacement>
    <DisplacementX>0</DisplacementX>
    <DisplacementY>25</DisplacementY>
  </Displacement>
  <Rotation>-45</Rotation>
</PointPlacement>
```

Opacity: Opacity is the transparency of either a label, point, polygon or line. It can be used to paint layers over each other (setting Opacity to 0), this is something used a lot incase multiple data has to be displayed in the same tile (used in our example as well). The way you use it is the following:

```
<Opacity>0.3</Opacity>
```

Rotation: Rotation is the function that is used to turn all shapes and labels in SLD. It is very handy if you want to turn tiles or make labels line up with lines better. The way to use it is very simple in the section that has to be

rotated just add the following code: `<Rotation>-45</Rotation>` for a negative 45 degree turn.

Graphic Fill: A graphic fill is used in case a picture/image has to be shown in a layer. It has a lot of possibilities since every picture/image can be added through this way. The implementation is a little more complex so below is an example of a graphic fill.

```
<FeatureTypeStyle>
  <Rule>
    <PolygonSymbolizer>
      <Fill>
        <GraphicFill>
          <Graphic>
            <ExternalGraphic>
              <OnlineResource
              xlink:type="simple"
              xlink:href="colorblocks.png" />
              <Format>image/png</Format>
            </ExternalGraphic>
            <Size>93</Size>
          </Graphic>
        </GraphicFill>
      </Fill>
    </PolygonSymbolizer>
  </Rule>
</FeatureTypeStyle>
```

There are a lot more options and a lot of the information can be found in the SLD cookbook on the GeoServer website. This section was meant to give some more insight the commonly used functions.

4.3 Client Side Development

Hier Mathijs will write a subsection on what is possible in terms of client side development with respect to the stairwalker project.

5 Running Example

Give an introduction to the example

5.1 Requirements

Before recreating the given example all necessary installations should be made and the required files obtained. The following programs should be installed:

1. PostgreSQL (Section 2.1.1)
2. PosgGIS for a PostgreSQL database (Section 2.1.3)
3. Tomcat or another web service server (Section 2.3)
4. GeoServer with extension (Section 2.4.3)

Also the following files should be at the ready:

1. The tool to make a pre-aggregate index: Pre-Aggregate-Index tool (Section 2.2.2)
2. Configuration file for the tool: `runningexample.config.xml`
3. Sample dataset: `RunningExample.csv`
4. Sample SLD file: `RunningExamplSLD.xml`

Currently the files can be found in the `RunningExample` directory in the same Git as this manual.

Furthermore it is useful to have a tool in which the PostgreSQL database can be managed. During development the tool pgAdmin[7] was used.

5.2 Example Table Setup

The first step of the process is to have a dataset which will be aggregated. For this example a dataset is supplied in the form of a `.csv` file. In order to import this first recreate the table in the database. This can be done with the following SQL query:

5.2.1 Recreating Dataset

```
CREATE TABLE runningexample (
        id_str character varying(25),
        tweet text,
        user_name text,
        place_name text,
```

[7] `http://www.pgadmin.org/`

```
        time timestamp with time zone,                          6
        reply_to text,                                          7
        place_id bigint,                                        8
        len bigint,                                             9
        coordinates geometry,                                   10
        CONSTRAINT enforce_dims_coordinates CHECK ((            11
           st_ndims(coordinates) = 2)),
        CONSTRAINT enforce_geotype_coordinates CHECK            12
           (((geometrytype(coordinates) = 'POINT'::
           text) OR (coordinates IS NULL))),
        CONSTRAINT enforce_srid_coordinates CHECK ((            13
           st_srid(coordinates) = 4326))
);                                                              14
```

Once the table is created import the `.csv` file to the table.

Note that `RunningExample.csv` contains column headers, uses `;` as column seperators and `"` as quote seperators. After the import is done a pre-aggregate index can be made.

5.2.2 Creating Pre-Aggregate Index

An in depth discussion of the use of the pre-aggregate index tool is given in Section 2.2.2. In the example only commands will be given with little explanation.

First the tool needs to be compiled which can be done with the command below executed in the `pre-aggregate-tools` directory.

```
mvn package appassembler:assemble                               1
```

The next step is to put the pre-aggreate tool config file in the `pre-aggregate-tools` directory. Once this is done the tool can be called with the following command. Note some variables will be different than in the listing below. These are `<database>`, `<host>`, `<port>`, `<pass>`, `<user>`, these should be filled out according how PostgreSQL was set up.

```
target\appassembler\bin\create-pa-index -config                 1
   runningexample.config.xml -d <database> -dbtype
   postgresql -h <host> -p <port> -password <pass> -s
   public -u <user>
```

This will create a pre-aggregate index of the dataset. In the database there will be three new tables. The pre-aggregate index named `runningexample_pa` and two help tables which keep track of the indexes and the axes used by those indexes. All the work on the side of the database is now done and the next step is to visualize the dataset using GeoServer.

5.3 GeoServer Setup

This section will give a step by step guide of how to create a visual geospacial representation using the pre-aggregated index of the example dataset. This will be done using GeoServer, specifically the GeoServer web administration interface. This section offers are concrete version of the deployment discussed in Section 3.

Figure 1 shows the `Data` section of the navigator which can be found on the left hand side of the web administration interface. The links in this section will be used to navigate between different pages needed to configure the whole setup.

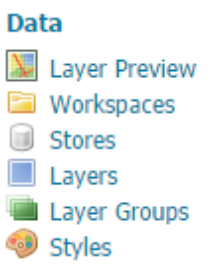
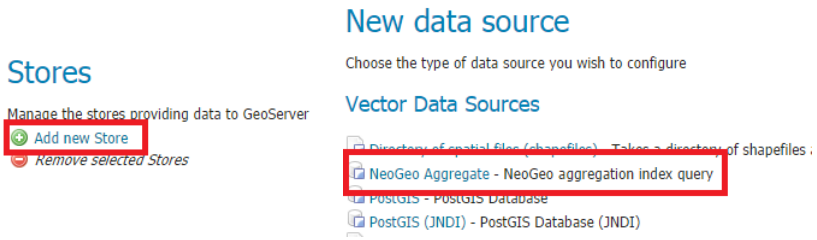


Figure 1: Data section of navigator

5.3.1 Add Source

A data source is added in the following manor:

1. Click on the `Stores` link in the `Data` section shown in Figure 1.
2. The `Stores` will open, the top of the page will look like Figure 2(a), click on the `Add new Store` link.
3. A selection of different `Vector Data Sources` is now available. Select `NeoGeo Aggregate` as shown in Figure 2(b).



(a) Add new store (b) Select data source type

Figure 2: Adding new `Store` to GeoServer

4. After selecting `NeoGeo Aggregate` as `Vector Data Source` a page like Figure 3 will open. Fill in all the fields as shown, some values may differ depending on how the database is setup. More exact information can be found in Section 3.1.
5. Once everything is filled out click the `Save` button. This leads to page where `Layers` can be published. However before that is done first the `Style` should be imported.

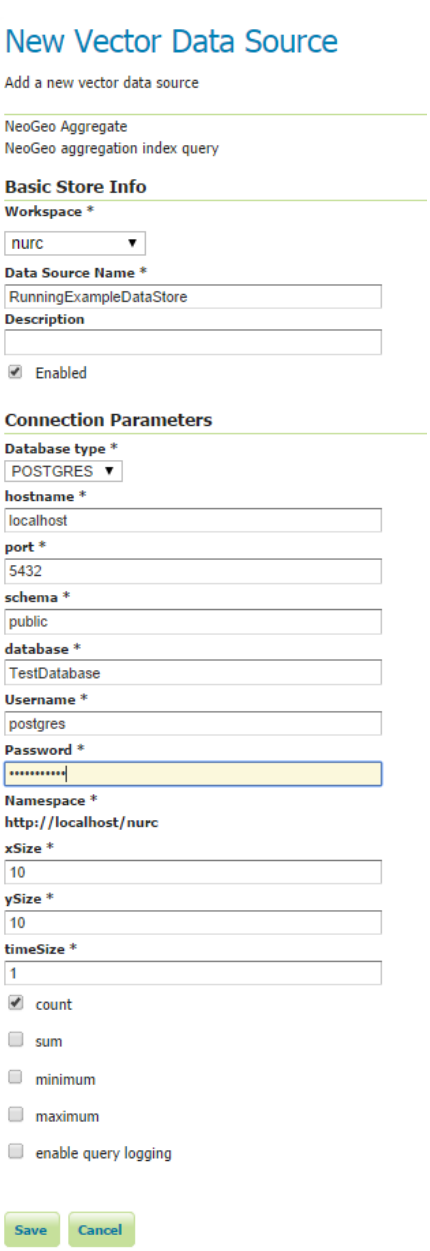


Figure 3: `New Vector Data Source`

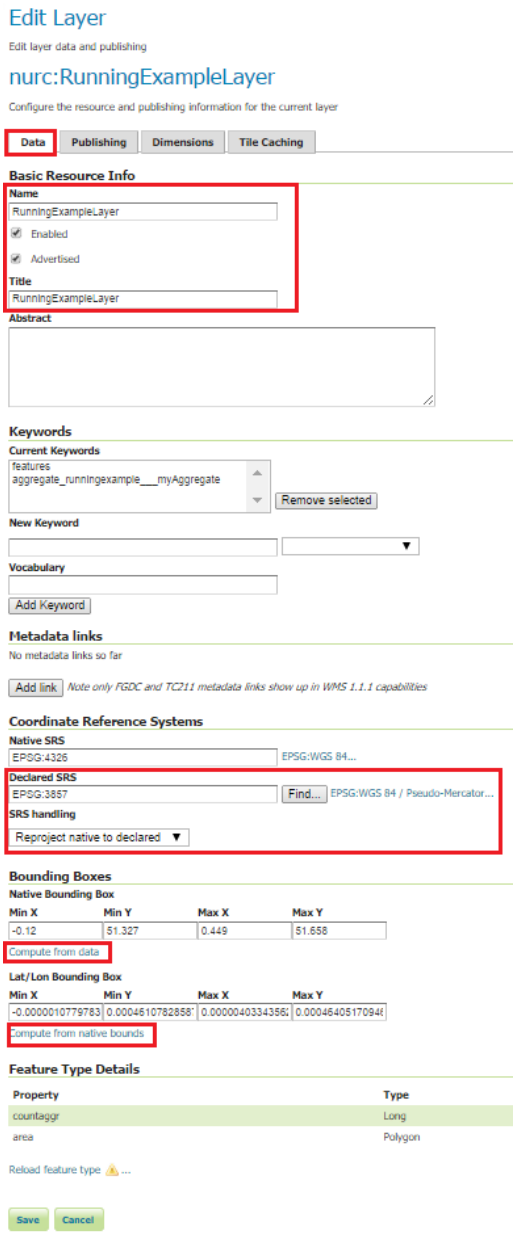


Figure 4: `Edit Layer` page

5.3.2 Import Style

Importing a style is done as follows:

6. Click on the `Styles` link in the `Data` section shown in Figure 1.
7. Click on the `Add a new style` button which will go a page similar to Figure 5 although empty.
8. Import the `RunningExamplSLD.xml` file by using the `Choose File` button then the `Upload...` link highlighted in red in Figure 5.
9. Once the style has been upload the `New style` page should look like Figure 5.
10. Press the `Save` button.

The style used in this example has now been imported in GeoServer and now the layer is ready to published.

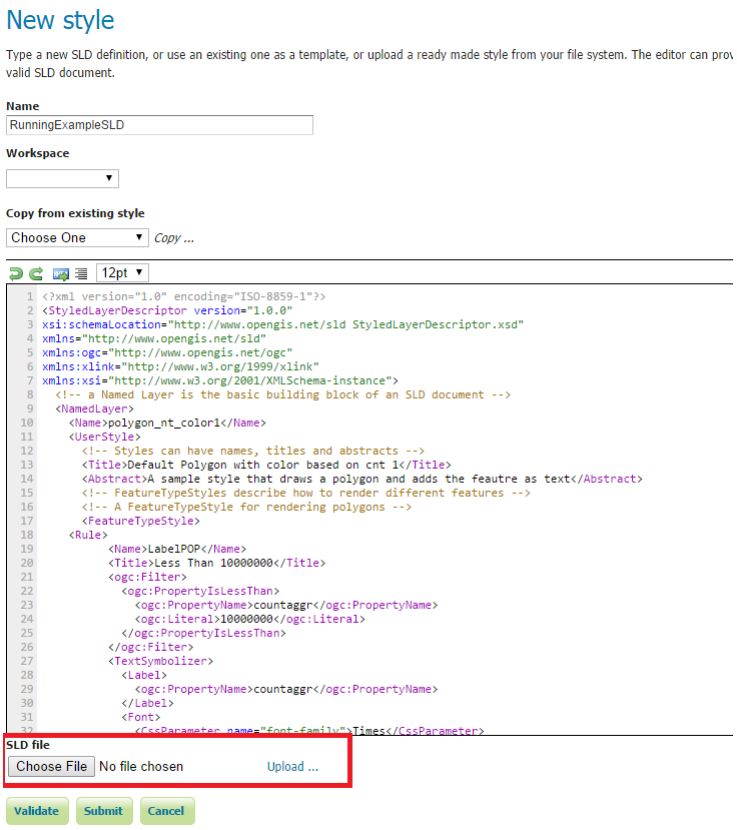


Figure 5: Importing SLD style from file

5.3.3 Create Layer

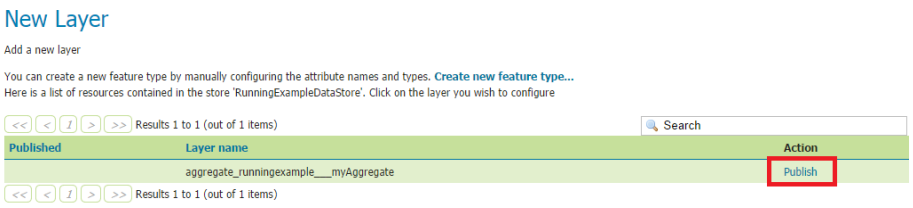


Figure 6: Publishing a `Layer`

Creating a new layer is done as follows:

11. Click on the `Layers` link in the `Data` section shown in Figure 1.
12. This opens the `Layers` page, here click on the `Add a new resource` button. This open a page similar to Figure 6.
13. Select the `Publish` action for the example layer.
14. A page like Figure 4 will open. Set highlighted fields to match Figure 4. More exact information about these fields can be found in Section 3.3.
15. After the fields in the `Data` are filled in, go to the `Publishing` tab which. See Figure 7.
16. Set the default style to `RunningExampleSLD` like in Figure 7.
17. Press the `Save` button.

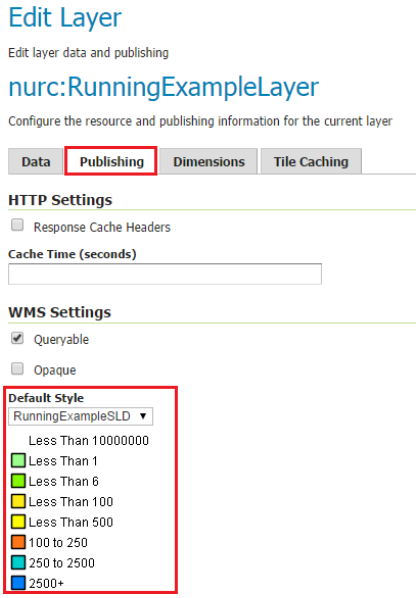


Figure 7: Adding a `Style` to the `Layer`

A layer for the example dataset has now been created and is ready to be viewed.

5.3.4 View Layer

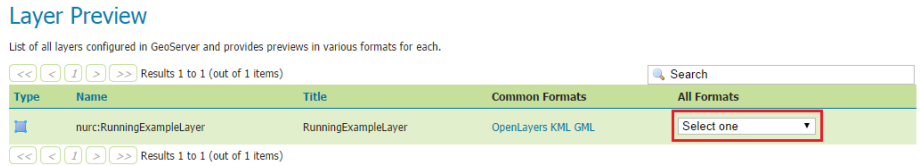


Figure 8: Previewing a `Layer`

The final GeoServer step is to preview the layer. The preview only shows the highest granularity of the aggregation index. Getting a preview of a layer is done as follows:

11. Click on the `Layer Preview` link in the `Data` section shown in Figure 1.
12. The `Layer Preview` page opens which displays all viewable layers like in Figure 8.
13. A preview format need to be selected from the drop-down menu highlighted in Figure 8.
14. Select a `WMS` format such as `PNG`.
15. A new web page will load (this might a few seconds depending on if the server side extension is enabled).
16. The final result will look like Figure 9.

The layer which shows the example dataset is now complete. The values of each square in the layer is calculated using the pre-aggregate index of the dataset. See 4.3 to learn how the layer can be used in combination with other tools such as OpenLayers[8] to a dynamic map which updates data on the fly.

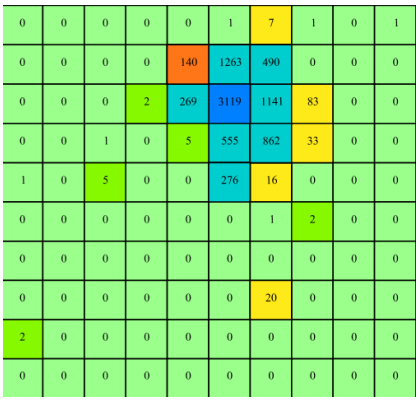


Figure 9: Preview of whole dataset

[8] `http://openlayers.org/`