BI Office

Mapping Framework & Installation Guide

Version 6.5

Contents

I. Overview

BI Office 6 includes 2 techniques for showing data on maps.

- The **first technique** is by flagging hierarchies and levels within a data model as a geospatial data type, and then having the BI Office engine find the matching shapes or geocoded location points and plotting them on a map.
- The **second technique** (new) is by striping your data model/cube with look up fields and then having the BI Office engine find the exact matching shape or data point and plotting it on a map.

Technique one is, quick and easy, but requires users to only use well known mapping entities (like countries). Although, custom shapes and lookup tables can be added via the administrative console.

Technique two is longer to setup and requires development work by model developers. However, it offers more accurate mapping with greater flexibility for unique and complex mapping scenarios.

This document will cover general setup and install of the geospatial database and configuration in the admin console. Then it will explain how to implement and use both techniques

II. Mapping Prerequisites

Clients wishing to deploy the Pyramid Geospatial solution will need to take up the following considerations:

- ***Space for the GIS database***: the Pyramid GIS database is currently **3GB** in size.
- ***Location name preparation***: Client geographic data should be prepared for the GIS geo-referencing.
  - Where possible, *Standard English* naming conventions should be used for all geographic entities. ISO-standard abbreviations can be used for countries and states.
  - Addresses provided should be constructed in a format that will allow the geocoding engines to successfully place the location on a map. The general format for a complete address is:

    ```
    Street number Street Name, City, State/Province, Country, Postal Code
    ```

- ***Licensing***: The Pyramid Mapping solution currently uses the Microsoft BING service for image tiles and geocoding services. Depending on your type of BI Office licensing, some clients may be required to provide their own Bing account and license key to enable and activate the mapping component.
  - For licensing go to http://www.microsoft.com/maps/product/licensing.aspx

III. Installation & Updates

A. Updating Existing Databases

If your company has already been using the BI Office mapping database in previous versions to 6 then all you need to do is just run the script **GisUpdate6.0.sql** (downloadable from the portal).

Open the script in SQL Server Management Studio and execute is against your existing BI Office database ("PyramidGIS").

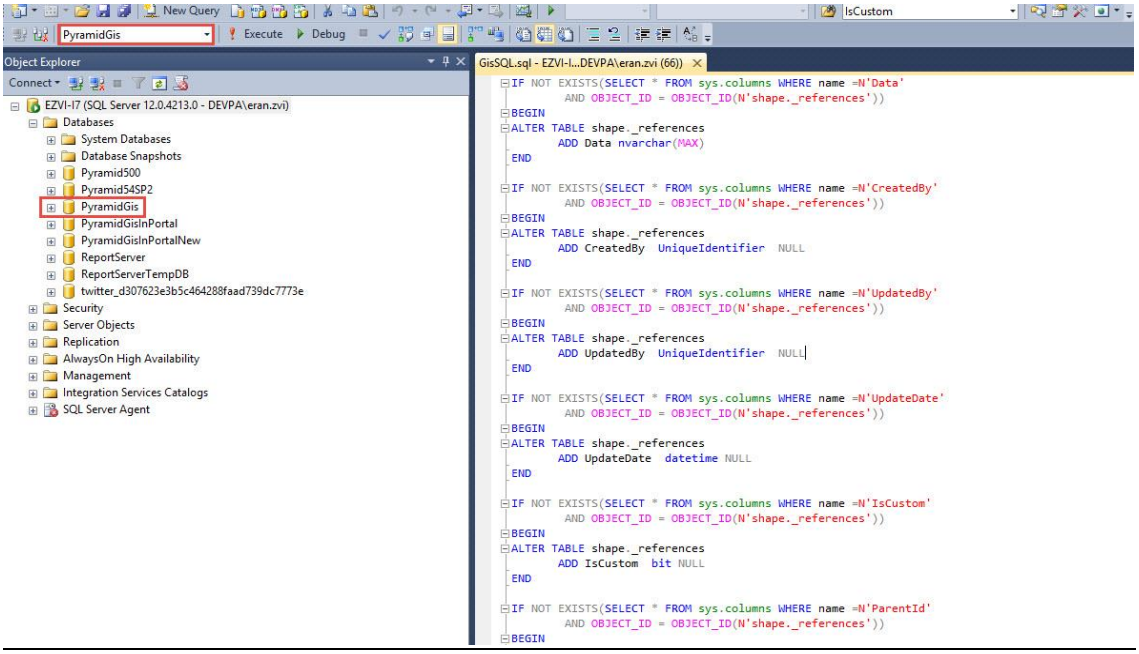


Figure 1

See the appendix for details on the changes made.

B. New Database Installations

- Download the Pyramid Geospatial database from the customer portal.
- Unzip the contents and copy "PyramidGis.Bak" to your local drive.
- Connect to the appropriate instance of the Microsoft SQL Server Database Engine.
- Right-click the database, point to **Tasks**, point to **Restore**, and then click **Database**, which opens the **Restore Database** dialog box.

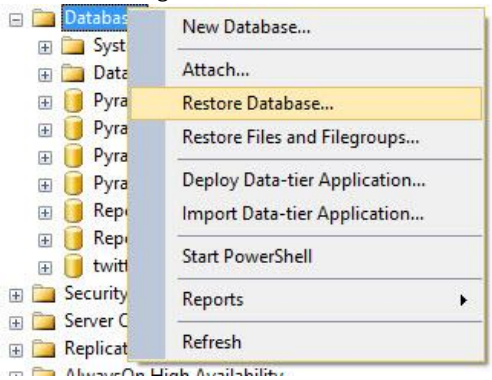


Figure 2

- On the Restore Database window choose the Device option and select the PyramidGis.bak file.

- Press the O.K. button and wait for a minute for the process to finish.

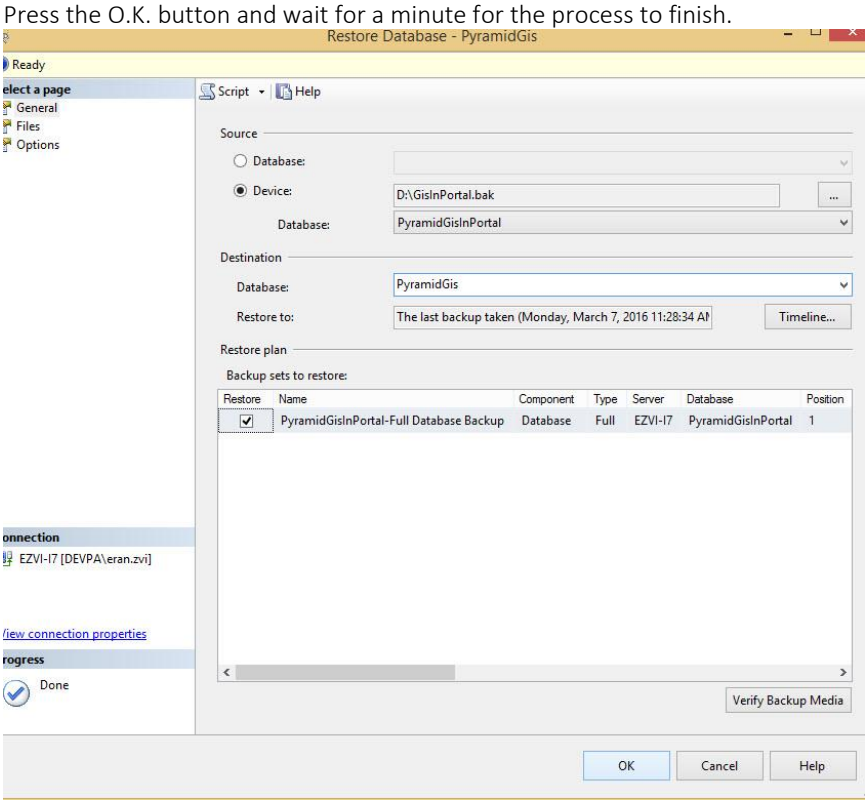


Figure 3

- SQL Server has now restored the Database and you can now go to Pyramid Admin to point the Pyramid Application to the Pyramid GIS Database.

i. Connecting the Pyramid GIS Database to the client

Once restored, enable Geospatial capabilities in BI Office:

- Inside the admin console, go to **Settings** tab and select the Geo Spatial option
- Complete the details as shown in the sample below.

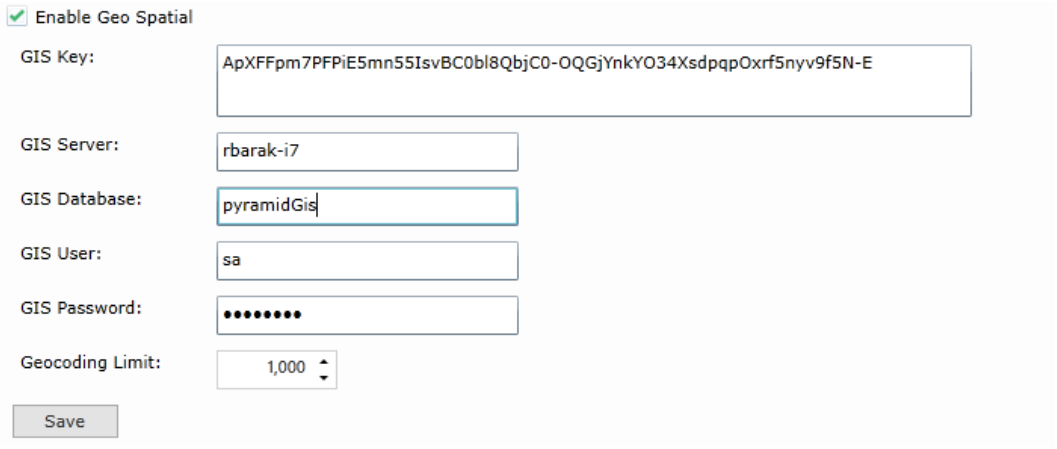


Figure 4

ii. Details on the fields

- **GIS Key**: Copy Paste the BING key supplied to you by Pyramid.
- **GIS Server:** The server that you have restored the DB to.
- **GIS Database:** The name of the Pyramid GIS Database.
- **GIS User & Password:** Your server credentials.
- **Geocoding Limits:** The Max amounts of Geocoding to perform per request.

NOTE: Once you are finished, click refresh and you will be able to see the list of table currently in your Database as in the image below

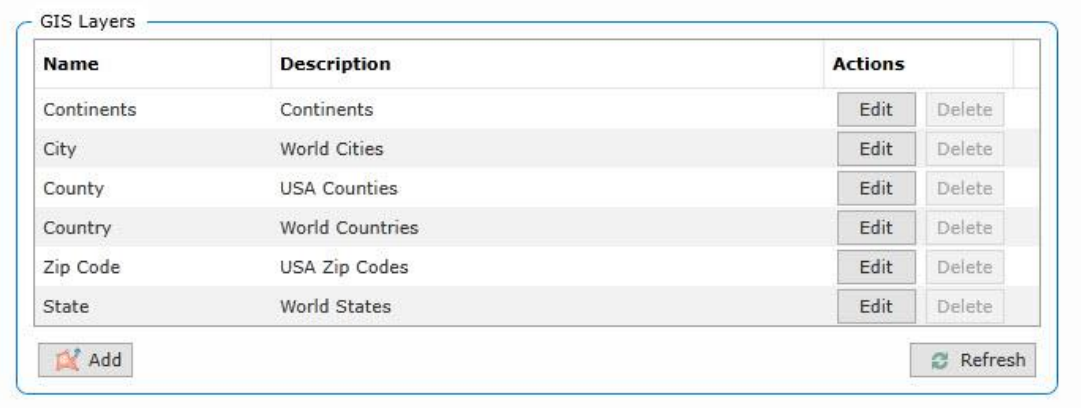


Figure 5

IV. Anatomy of the Pyramid Geospatial database

By default, the Pyramid GIS database is installed under the name '*PyramidGIS*'. The database contains several **lookup** tables (contained in the standard *dbo* schema) and a separate **shapes** table (contained in the separate *shapes* schema).

The current version of the *PyramidGIS* database includes the following lookup tables:

- World Continents
- World Countries
- World States/Provinces[1]
- World Cities
- US Counties
- US Zip Codes

Each table has a variety of name fields which can be used by developers in the following *matching* process.

Client developers use the lookup tables in the *PyramidGIS* database to match up against their own geographic fields in their databases, to map the shape **geospatial** lookup keys. These keys are GUID values in fields called "PAID" - short for "Pyramid Analytics ID". These fields are used by the analytics engine to find the matching shapes in the geospatial shapes table.

Often, geographic data is presented in hierarchies from country to state to county to city to zip/post code. As such, it is expected that a given geographic table will contain multiple *PAID* keys that are used in the cube's hierarchies.

Where no matching shape file is available or appropriate, developers can also provide an 'address' that will be **geocoded** by the Pyramid application as needed. When supplying an address, developers should try and provide the most complete address possible in the member's address properties, so geocoding is successful and accurate.

These processes are described in more detail below.

V. New GIS Admin & Upload Mechanism

C. GIS Admin

The new GIS upload mechanism comes to replace the old standalone tool. The upload mechanism has two functionalities, whether to load shape files or to load addresses that will be Geocoded and will be inserted to the GIS DB as a point geometry (the benefit of doing this is that each time an address will be requested no geocoding will be done).

In order to facilitate the possibility to work with Tabular models we have added functionality to designate relationship between columns in the DB.

Example: If we take the states table (WORLDstates in the Pyramid GIS) and we look at the data contained in it, then one can associate the relationship based on the Name & Country columns. (Figure 6)

[1] States and Provinces as defined as "Level 1" administrative detail level by the UN.

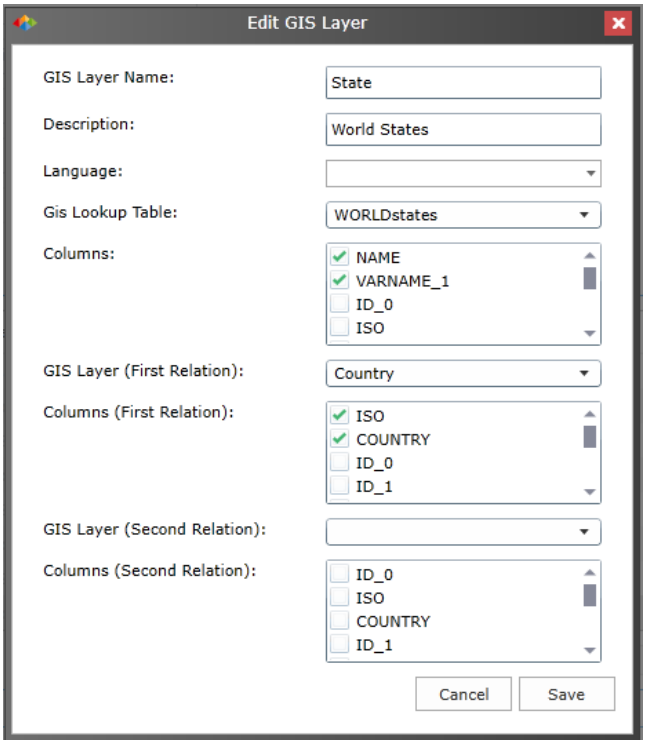


Figure 6

In figure six you can see that the first look up columns are Name & Varname_1. So each state name will be searched upon this two columns. Also you can noticed that we designated two First Relation columns (and secondary as well). As a result if your grid results contain the State "Montana" on the rows or the columns & the parent of Montana is the United States then the map will show Montana in the United States. If one will omit the First Relationship and will deselect the associated checkboxes then the query might return a different Montana then the one in the United States since Montana is also a state which belongs to the Country of Bulgaria.

Figure seven shows the data in the States table when one select a state Called Montana and emphasizes the need for the First Layer relationship.

PAID	ID_0	ISO	COUNTRY	ID_1	NAME	VARNAME_1	NL_NAME_1	HASC_1	CC_1
2AF9B8D4-947F-4E61-ADF3-0FC812CBE39C	24	BGR	Bulgaria	340	Montana	Mikhaylovgrad|Mihailovgrad	???????	BG.MT	
CD250B73-8997-4552-81DE-E18948A18C0E	234	USA	United States	3217	Montana	MT		US.MT	

Figure 7

D. New Upload Mechanism

iii. Adding shapefiles

The Geo Spatial Admin section also encapsulate in it the possibility to load shapefiles (WGS 84) & Addresses to the Pyramid GIS. Working through an example will explain this feature best.

If we want to upload the counties of Australia, we would first prepare our shapefile .ZIP file.

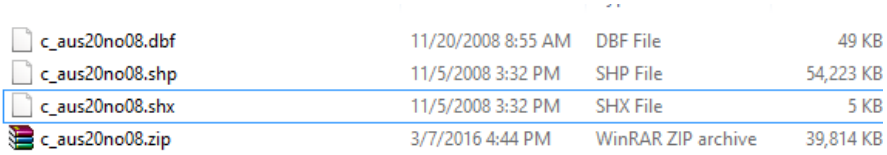


Figure 8

Note that minimum three files are required for the shape file uploader to succeed. The shapefile.**dbf,** shapefile.**shp &** shapefile.**shx.** Must be all zipped together under one file. After filling the required information for the layer name (logical name that will appear in the pyramid client), the description, the target table name and pressing next, the user will be present with the same as in figure nine

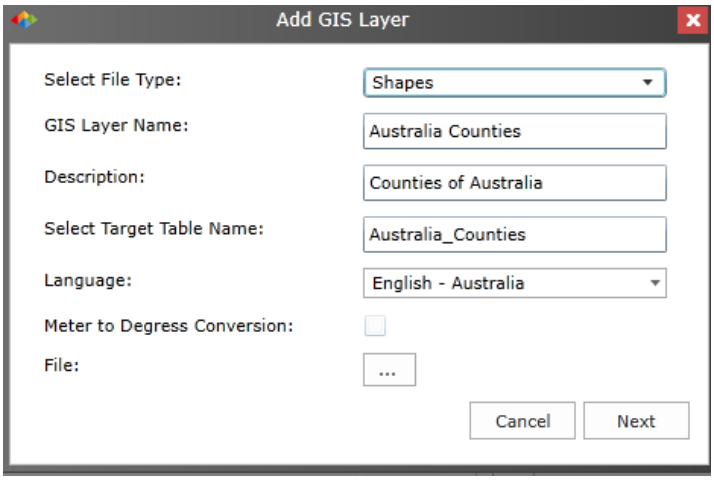


Figure 9

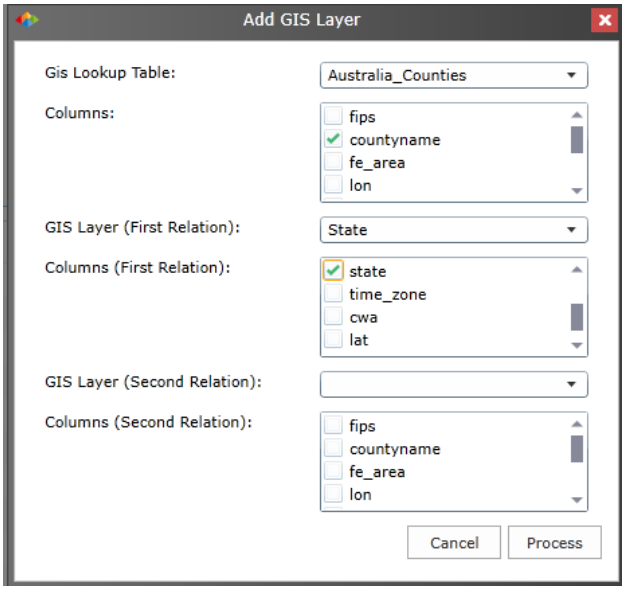


Figure 10

Pressing the process button will cause the shapefile to be uploaded and processed. (Please note that this process may take up to several minutes).

If for example one created a Mash Data source from the table created he can then designate the county name as a Geo Hierarchy (figure 10), and choose a subset of the data that will result in the mapping to show the following shapes.

Figure 11

iv. Adding addresses

The loader also supports the load up of addresses to the pyramid GIS through a text file. This addresses would be Geocoded through the BING API and would be stored in the Pyramid GIS as Latitude Longitude points. Please note that the text file must have the address column in it and. The file must be tab separated.

```
ADDRESS
Matanuska Sustina AK USA
Prince Wales Hyder AK USA
Wrangell AK USA
```

VI. Technique One

E. Overview

The broad steps to deploying the Pyramid mapping component are as follows (see figure 12 below):

1. **The client installs the Pyramid SQL Server 2008 geospatial database ("PyramidGIS").** The database currently provides the geographic reference points for world countries, states[2] and cities as well as US counties and US zip codes.
2. **The client maps the geographic reference points in the PyramidGIS to their own dimensional data in their own relational databases (where possible).** Mapping of such data is completed through the use of various ISO-standard naming keys or the plain English names of the entities.
3. **The client adds the new geographic reference keys into their SSAS cube designs.** Keys, created in step 2 above, are added as attributes that will act as *member properties* for existing geographic items in their current dimension(s). Newly added attributes are flagged as geospatial attribute properties using the TYPE setting.
4. **Addresses and other geographic entities that are NOT reflected in the PyramidGIS are designated as "location-address" attribute types by the client.** This will ensure that the Pyramid engine attempts to geocode some data points and plot them on a map where possible.
5. **End users can now analyze their data geospatially in the Pyramid application**. End users are able to browse the cube and slice and dice their data as normal in the Pyramid Application. If the user queries a geospatially flagged hierarchy, the mapping component and map ribbon tab are enabled and data can be plotted on a map.
    a. In plotting data the application will attempt to either:
        i. Use the pre-coded geospatial reference keys (prepared in step 3 above) to retrieve the appropriate geospatial objects from the **PyramidGIS** database and display them in the mapping component.
            1. Boundary reference points will be exhibited as color-coded 2-dimensional shapes or multi-colored plotted pie charts depending on their user's preferences and the underlying query used.
            2. Users can elect to view plotted data on top of mapping tiles using graphic or satellite imagery or against a plain skeletal world map.
        ii. Geocode the location addresses without a pre-coded geospatial reference key and plot them in the mapping component.
            1. All successfully geocoded addresses will be shown as multi-colored push-pin pie charts
            2. Users can elect to view plotted data on top of mapping tiles using graphic or satellite imagery or against a plain skeletal world map.
    b. All data points within the mapping component will provide context menus such as drill, dice, actions and isolate/exclude transactions. Where possible and appropriate, tool-tips with entity data will also be provided.

[2] States and Provinces as defined as "Level 1" administrative detail level by the UN.

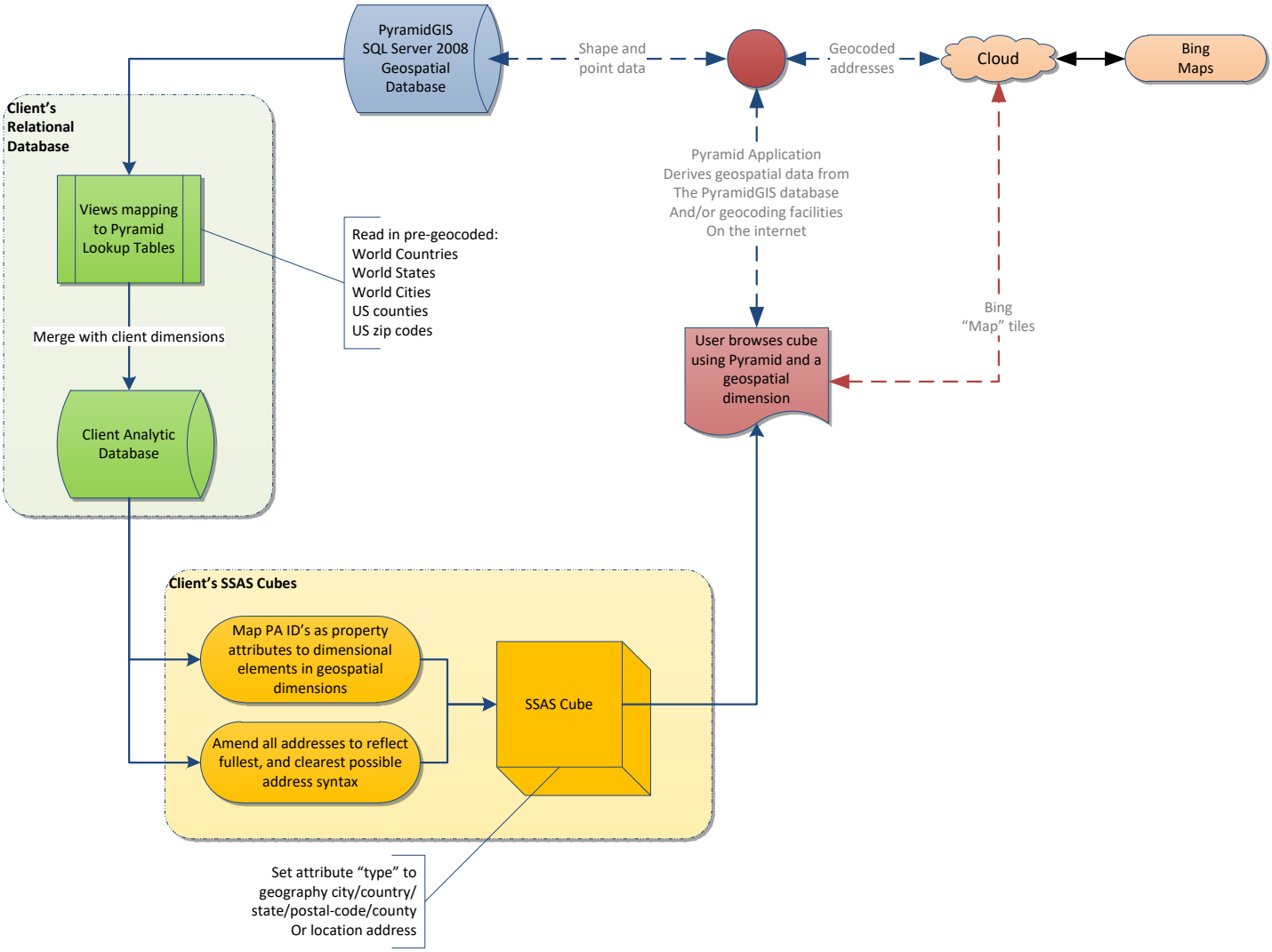


Figure 12 Pyramid Geospatial Flow

F. Geospatial mapping to Client Data

The following steps are *recommended* for fusing the geospatial reference keys from the Pyramid GIS database into client dimensional data.

v. Relational Database Integration

1. Developers create views in the target client relational database that simply read the lookup tables from the source '*PyramidGIS*' database. (If the *PyramidGIS* database is installed on a separate database server, the PyramidGIS database server should be linked first).
   Example T-SQL for creating a view linked to the *PyramidGIS* database is presented below. Here, the '*UScounties*' lookup table is being read into the client's relational database as a view called '*gisCounties*'

```
CREATE VIEW [dbo].[gisCounties]
AS
SELECT      *
FROM        PyramidGIS.dbo.UScounties
```

2. Once the relevant lookup tables have been mapped into views on the client's relational database, developers should then endeavor to map the GIS foreign key(s) into their dimension tables. The foreign key field is always designated as the *PAID* column, of type '*uniqueIdentifier*'. Several standardized name and lookup fields are presented on the Pyramid lookup tables to facilitate this cross-walking exercise.

   If a dimension needs to have multiple lookups (say for country, state/province, county, zip/postal code, city), then EACH of the *PAID* foreign keys needs to appear on the dimension table. Each foreign key field should be given a unique name.

   Basic example T-SQL code is presented below to map the *PAID* key from the gisCounties view to an example '*DimCustomer*' dimensional table as a field called '*countyPAID*'. Notice that the join used is an **OUTER** join (so records in the dimension that do NOT match the lookup are not dropped in the resulting view). Also in this specific case, US counties often have the same name within different states. As such, two joins are required to ensure that the match against the dimensional table produces unique values – one for the county name and one for the state/province.

```
CREATE VIEW [dbo].[vDimCustomer]
AS
SELECT      gislook.PAID AS countyPAID, cust.customKey,
            cust.FirstName, cust.LastName,
            cust.Address, cust.City, cust.County, cust.State,
            cust.ZIP, cust.Phone, cust.Email,
            cust.Country, cust.ISO, cust.State_Province

FROM        dbo.gisCounties AS gislook
            RIGHT OUTER JOIN
            dbo.DimCustomer AS cust
            ON gislook.STATE = cust.State_Province
            AND gislook.NAME = cust.County
```

A more advanced example SQL statement is provided below that maps multiple *PAID* foreign lookup keys from various GIS lookup tables to a single dimensional table in a client's relational database. Notice that some table joins require two or three join keys and use different fields to make the match possible.

```
CREATE VIEW [dbo].[vDimCustomer]
AS
SELECT
      country.PAID AS countryPAID,
      state.PAID AS statePAID,
      city.PAID AS cityPAID,
      county.PAID AS countyPAID,
      zip.PAID AS zipPAID,
      cust.*
FROM
      dbo.gisCounties AS county
      RIGHT OUTER JOIN
      dbo.DimCustomer AS cust
            ON county.STATE = cust.State_Province
            AND county.NAME = cust.County
      LEFT OUTER JOIN
      dbo.gisPostCodes AS zip
            ON cust.Country = zip.COUNTRY
            AND cust.ZIP = zip.NAME
      LEFT OUTER JOIN
      dbo.gisCities AS city
            ON cust.ISO = city.ISO
            AND cust.City = city.AccentCity
            AND cust.State_Province = city.STATE_PROVINCE
      LEFT OUTER JOIN
      dbo.gisStates AS state
            ON cust.State_Province = state.NAME
            AND cust.Country = state.COUNTRY
      LEFT OUTER JOIN
            dbo.gisCountries AS country
            ON cust.ISO = country.ISO_2DIGIT
```

vi. Cube Dimension Integration

3. Once the client relational database has been setup and the *PAID* fields have been integrated into the dimension tables, the new fields need to be mapped into the relevant cube dimension(s). Open up the cube in Business Intelligence Development Studio (BIDS). Then open up the relevant *data source view* and click the refresh button to import the new PAID fields into the schema. An example using Adventure Works is provided below.

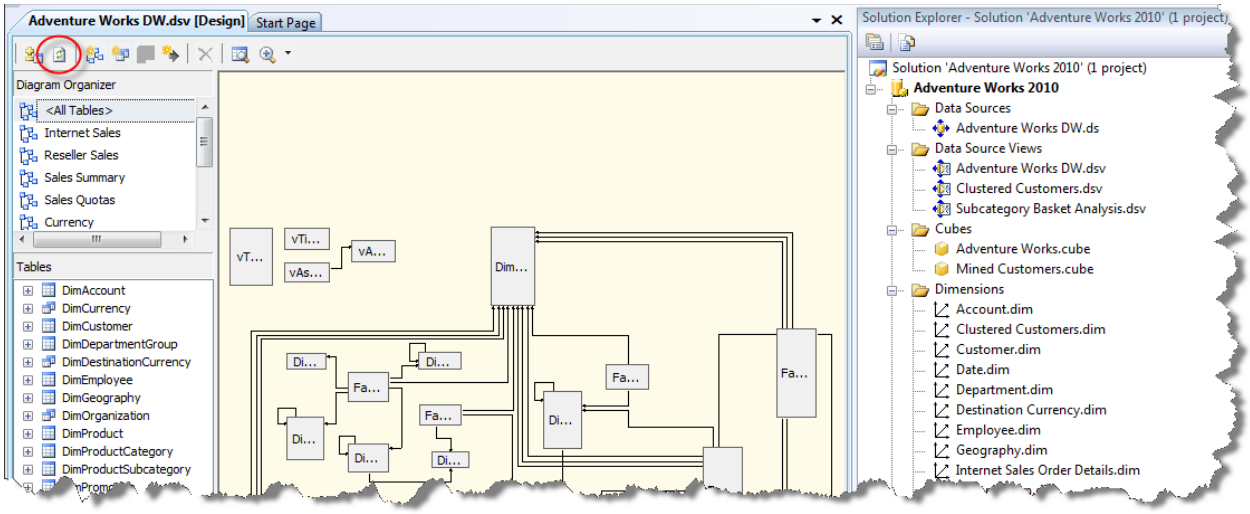


Figure 13 Cube Data Source View Refresh

4. Next, open the relevant dimension and drag / add the new *PAID* fields to the list of attribute hierarchies on the left. (In the example below three *PAID* fields are added to the Adventure Works Customer dimension: *PostCodePAID*, *StatePAID*, *CountryPAID*)

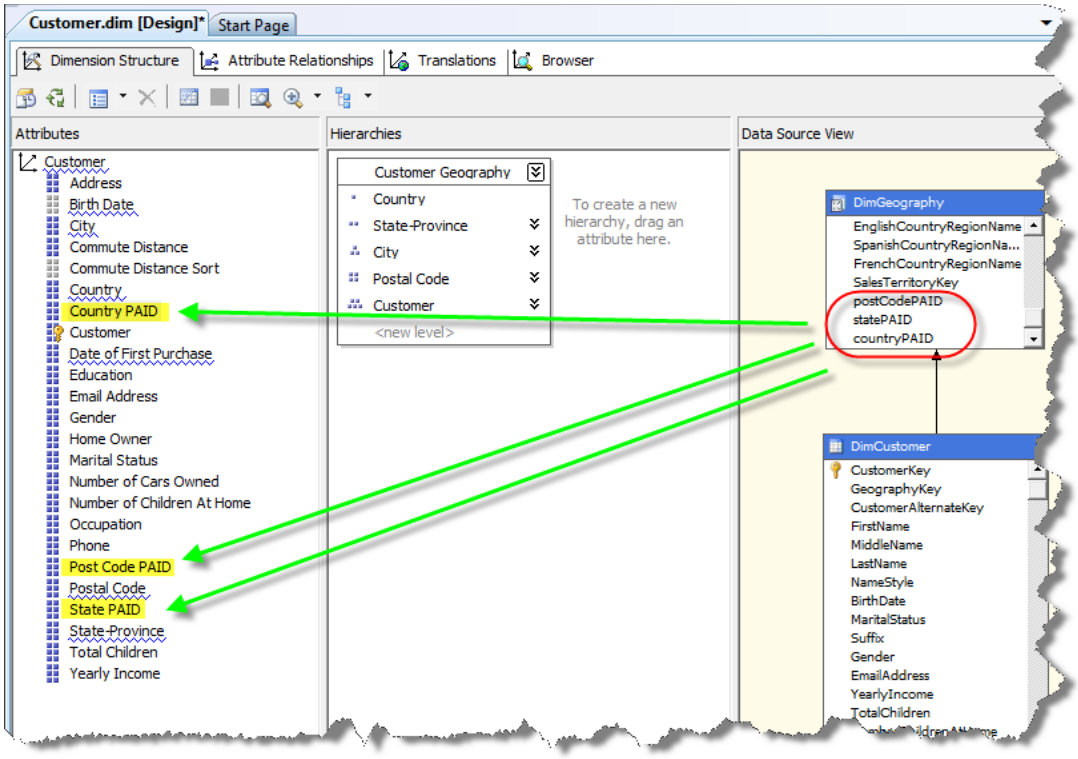


Figure 14 Adding the PAID fields as Attribute Hierarchies

5. For each of the new **GEOSPATIAL** attribute hierarchies added, change their properties accordingly (see figure 14 below):

a) Set the *AttributehierarchyEnabled* to False
b) Change the *Type* setting to ***GeoBoundaryPolygon*** type within the **Geography** section

6. Developers can also provide a complete address attribute for geographic fields that cannot be matched to a geospatial key field in the *PyramidGIS* database. Like the *PAID* keys, a separate 'full address' field should be added as a new attribute to the dimension's attribute hierarchy list, with the following property changes:
   a) Set the *AttributehierarchyEnabled* to False
   b) Change the *Type* setting to the ***address*** type within the **location** section.
7. After adding and amending the attribute hierarchies, it is CRITICAL that developers set the appropriate *attribute relationships*. For each attribute ensure that it is set as the *related* attribute to its counterpart *source* attribute. As shown in figure 15 - 17 below, we have made the *Post code PAID* attribute related to the *Postal Code* source attribute. This step ensures that the new attributes are viewed as member properties in the resulting cube structure for the selected members.

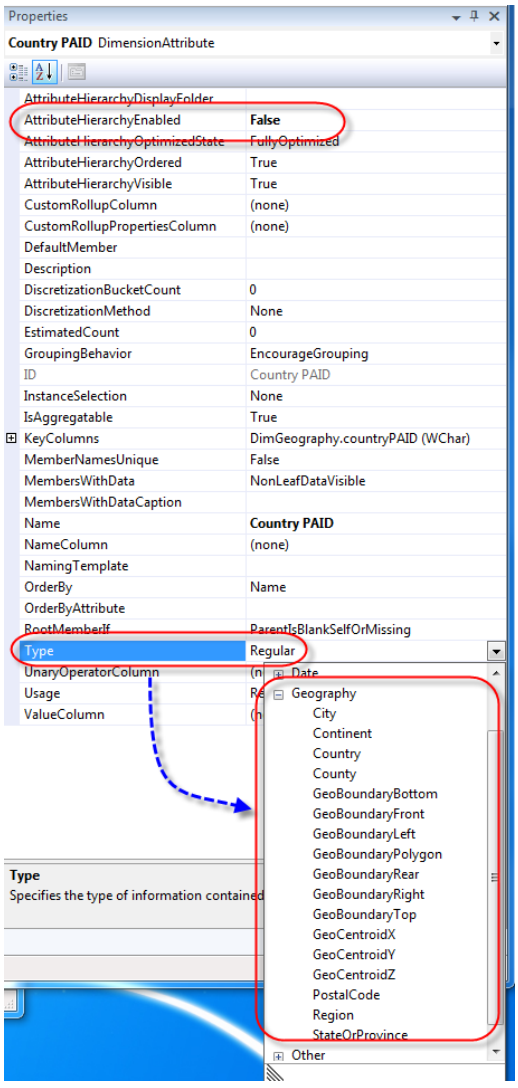


Figure 15 Attribute Hierarchy Properties

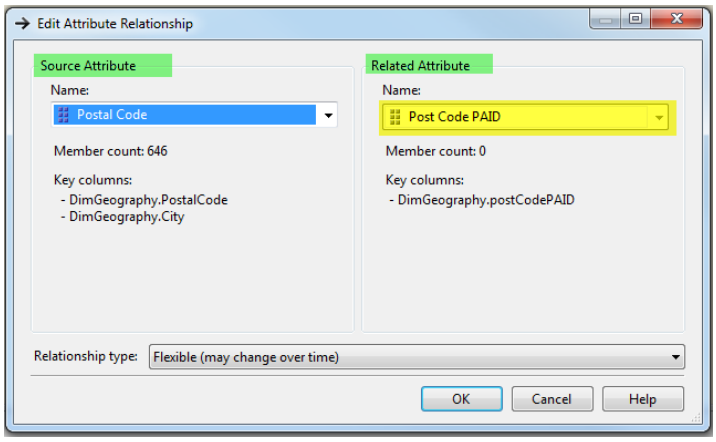


Figure 16 Setting Attribute Relationships

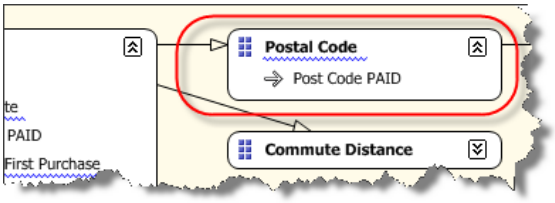


Figure 17 Attribute Relationship

8. The dimension and the cube can then be processed and checked within a cube viewer application (either BIDS browser, Management Studio or Pyramid itself). When looking at a geographic element's properties, the *PAID* unique identifier values should be visible as a member property attached to each hierarchy member (see figure 18 below).

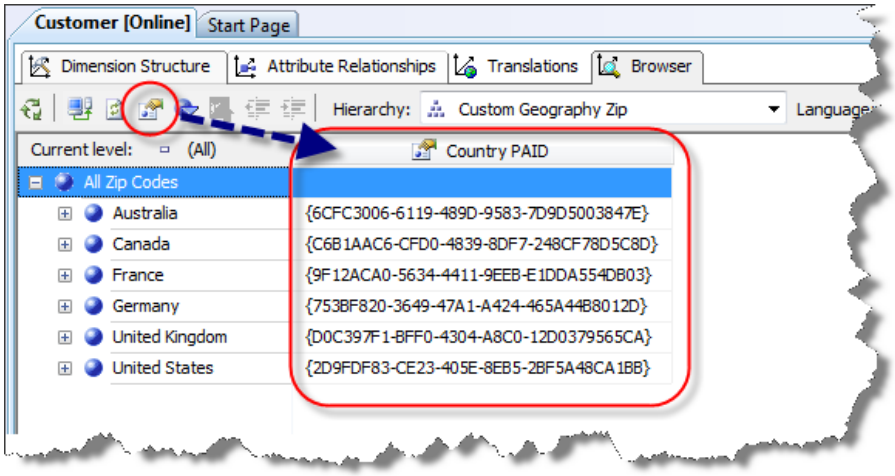


Figure 18 Viewing the PAID GUID values in the BIDS dimension browser

G. Using the Geospatial Mapping in the Client

Once the above steps have been completed successfully, the geospatial capabilities in the Pyramid Analytics client will be accessible (with the appropriate licensing).

The map report component and the map ribbon tab will be enabled and made visible if a user elects to include a geospatial hierarchy in their query (see figure 19 below).

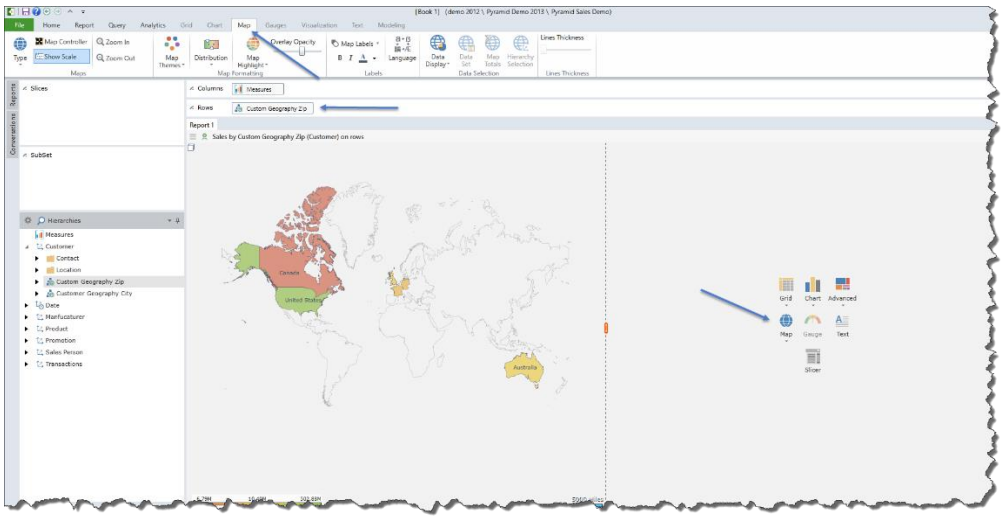


Figure 19 Mapping Components in the Pyramid Client

As the user drill downs into the geospatial hierarchy, the client engine will attempt to retrieve the matching shapes to the elements selected in the query (using the selected *PAID* keys) and plot them on the map in the client browser. Shapes will be changed to pie charts if multiple data points are presented for each location.

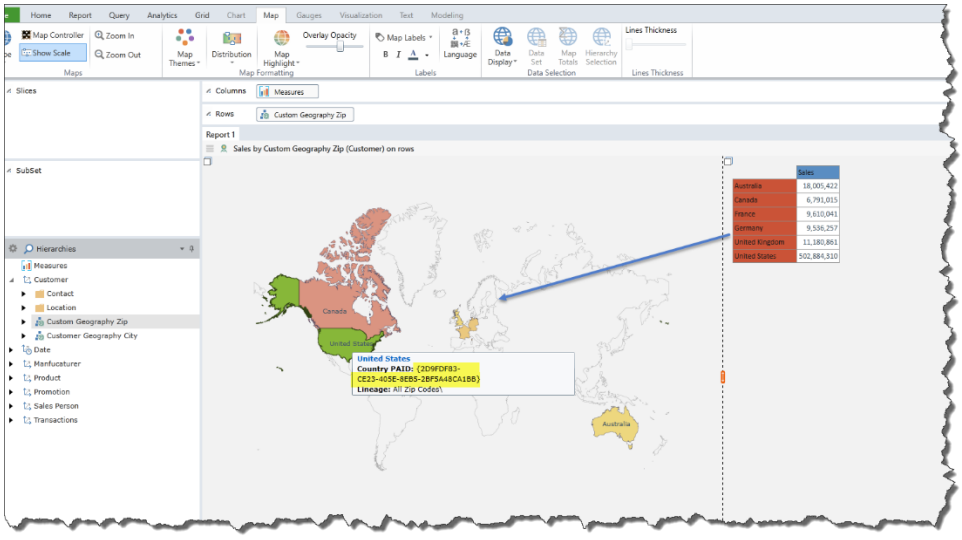


Figure 20 Cube data plotted onto maps

When no *PAID* key is found, the Pyramid Mapping engine will attempt to plot any members using member properties designated as 'location /addresses' types. These addresses are geocoded and then plotted as push-pins onto a tiled map (see figure 21 below). Geocoding of addresses is best completed when address fields are well structured and complete. As described previously above, the general format for an address should be:

`Street number Street Name, City, State/Province, Country, Postal Code`

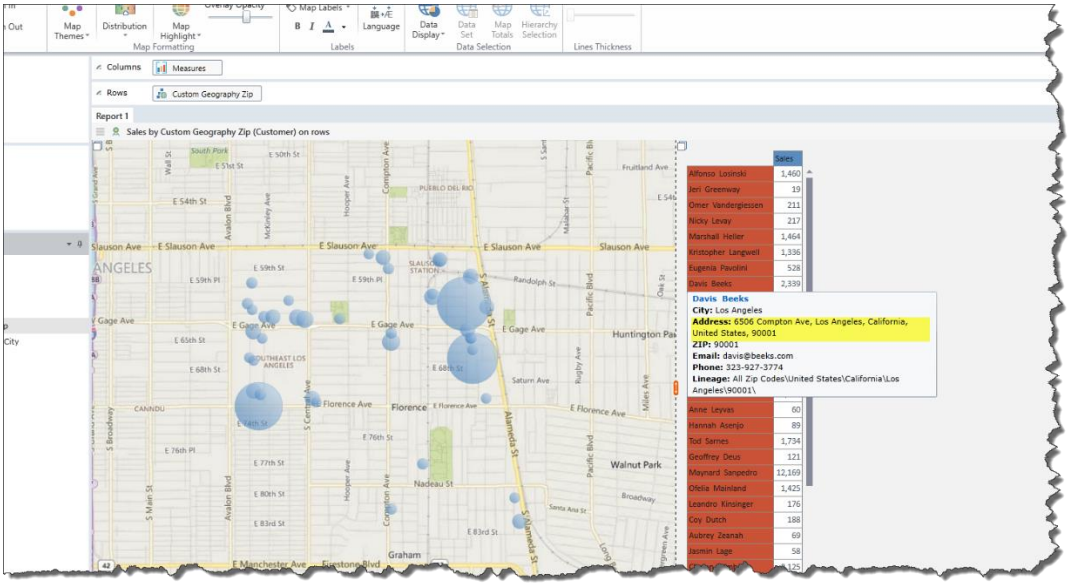


Figure 2 Plotting of Geo

VII. Technique Two

H. Overview

Version 6.0 introduces the ability to use the geospatial mapping without needing to use the 'PAIDs' integration technique as described in the previous section.

This technique is quicker and easier, but offers slightly less flexibility and accuracy in some situations - depending on your data, data models and requirements.

The process involves users flagging hierarchies (and levels) as geospatial content from inside the data discovery tool. In flagging the content, the user picks which geospatial data type the target is. This is then used to look up shapes in the database for the visuals. If the geolocation details are missing, a sophisticated interpolation engine is used to determine which shape to retrieve.

I. Integrating Technique 1 and 2

If a user queries a geospatially flagged hierarchy and the data source does not contain the PAIDs integration, the BI Office mapping engine will perform a textual search on the data in order to fetch the appropriate geospatial entity from the database (a polygon shape, a point of interest, etc.). This uses both the PAID technique and the new generic lookup technique combined.

vii. Setting Geospatial attributes settings.

If a data model (OLAP or tabular) was created and the without (geospatial) hierarchy types during its creation, users can still get the mapping capabilities by setting hierarchy types in the data discovery client. Users need to right click on a hierarchy in the hierarchy tree component and choose the appropriate geospatial data type to be mapped to. Doing so will set the non-Geographical hierarchy to a Geo one as seen in figure 24.

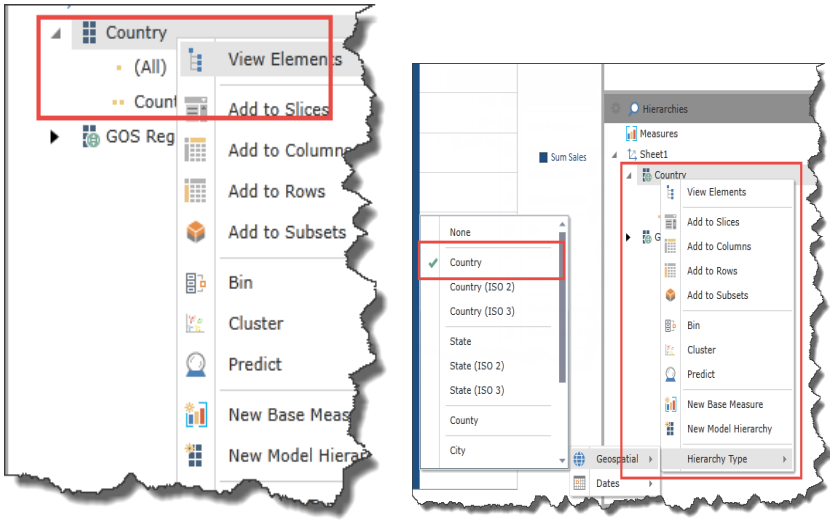


Figure 24

viii. Custom Shapes Mapping

It is also possible to map Geographical hierarchies to custom geospatial types (tables) that have been uploaded into the Pyramid GIS database via the administrative upload tool.

For example, if we look at a model that was uploaded via the New Data Model wizard that contains all Australian counties one can open the model and set the counties to “countyname” hierarchy and the BI Office mapping engine will commit the textual search against the custom table Australia_Counties that was created.

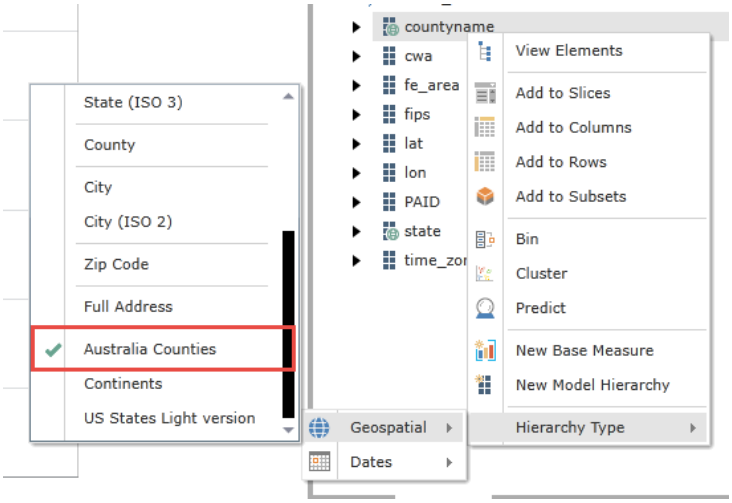
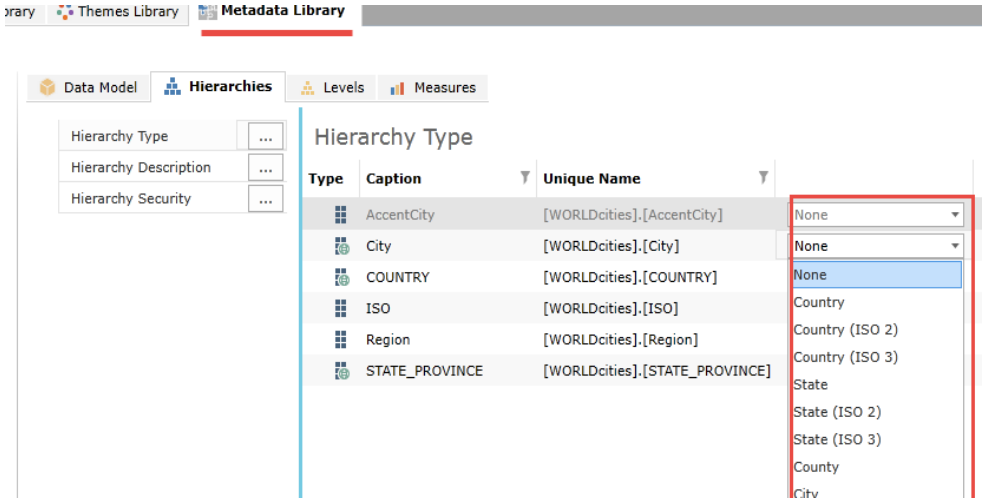


Figure 25

Figure 26

It is also possible to set hierarchy types in the Admin console in the Metadata editor (under Content). After you set the attributes they will be used globally in your organization



J. Data Model Setup

When creating a new data model in BI Office, users can pre-set the geospatial data types for each attribute – as shown in figure 22 below. When opened, the model will use the default Pyramid GIS geospatial data types and will return the appropriate shapes based on the names.

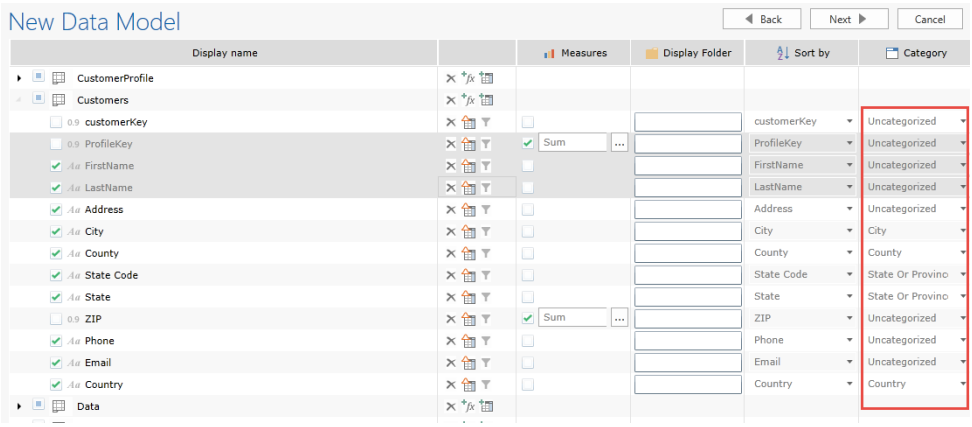


Figure 3

When the model is opened, these hierarchies are marked with the geospatial icon (figure 23).

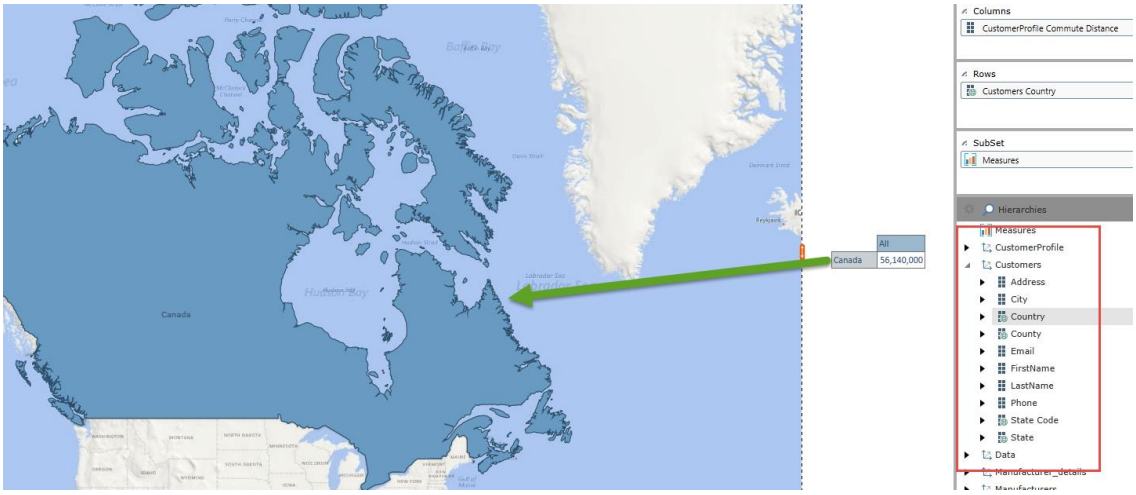


Figure 23

VIII. Appendix

K. Proxy Server Configurations

The Pyramid Mapping solution uses an Internet Access on the client browser and on the Application Server. In case a proxy server is set on the LAN as a gateway for an Internet Connection it should be configured by providing the address of the proxy server in the configuration wizard.

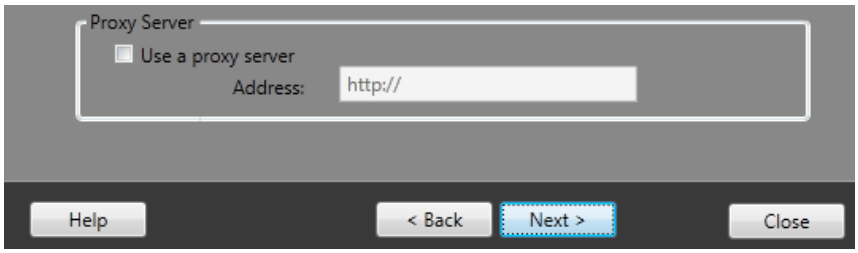


Figure 4 GIS Installer Proxy Server Settings

These 2 settings can be MANUALLY changed in “Pyramid.Satellite.exe.config” and “Pyramid.Server.Application.exe.config” files found in the Pyramid installation directory (see below). The GIS configuration tool will automatically set these.

On the “appSettings” node there are two keys:

- **EnableProxyServer** – set to true
- **ProxyServerAddress** – set to the proxy server address as per above

```
<appSettings>
  <add key="MaxThreads" value="50"/>
  <add key="ServerType" value="1"/>
  <add key="WorkspaceID" value="2"/>
  <add key="ServiceName" value="Pyramid_AppServer1
  <add key="ClientSettingsProvider.ServiceUri" va
  <add key="SQLConnectionString" value="Data Sour
  <add key="TopMru" value="10"/>
  <add key="CommandsDir" value="Services"/>
  <add key="LogonLocaly" value="true"/>
  <add key="EnableProxyServer" value="false"/>
  <add key="ProxyServerAddress" value="http://"/>
</appSettings>
```

Figure 27 – Configuration files Application Settings Keys

L. Uploading Custom Shape files and Addresses

Installed separately with the GIS database is the Pyramid "GEO" tool that allows users to upload their own custom shape files, global points list (using latitude and longitude) and address list (using addresses that will be geocoded).

The tool is installed on the same machine as the GIS database. Use the help provided with that tool for understanding how to add/edit/delete geospatial content in the Pyramid database.

ix. Custom Shapes

Using 3rd party tools, it's possible for users to create their own geographic ESRI shapes – representing customized regions that reflect their unique geospatial analytic requirements. Using the GEO tool, these shapes can be uploaded into the database and then referenced from the relevant cubes (as described elsewhere in this document). On using a cube, users will now be able to analyze data by these unique regions.

x. Custom Points and Addresses

If users have unique global points they wish to analyze, they can upload them into the GIS database and then referenced from the relevant cube(s). On using those cubes, users will now be able to analyze data by these unique geospatial points. The points can be uploaded using latitude and longitude positions or via a bona-fide address. Addresses are geocoded before loading them into the database.

M. Database Changes in Version 6.0

The shape._reference table was altered in order to accommodate the changes to the logic of tabular mapping. (Please note that if your company was already using a previous version of the Pyramid GIS you **must** run the upgrade script). Each row in this table designates a table in the GIS (Both custom table that were loaded via the tool & default tables)

The following is the schema for the shape._reference table:

- **Id** – a unique key the distinguish records in the shapes table (RefId column).
- **Name** – Name of the table.
- **Description** – description of the table.
- **CreateDate** - Create Date of the table
- **Data –** XML data the holds information regarding the relationship between the tables.
- **CreateBy –** User that created the table.
- **UpdateBy –** User that updated the table.
- **IsCustom** – A Boolean that designates whether the table is a default one or was uploaded via the tool (Default table are not able to be deleted).
- **ParentId –** A key used internally to designates relationship between tables.
- **GrandParentId -** A key used internally to designates relationship between tables.
- **RefTableName –** The table name as it appears in the DB.