Intersection Engine upgrade use cases

Contents

Overview

The GeoSpatial Intersection Engine (aka Dynamic Geographic Query Engine) is a tool for generating spatial intersections across geographical layers.

Background

- See the original Use Case UML for Dynamic Geographic Queries on the FI Spatial Infrastructure
- Check out the related Use Cases for dynamic geographic queries

Technical Requirements

1. Use a different web framework. The current implementation relies on Wicket and Ext. Wicket generates URLs which are not compatible with the FAO web servers security filters. Moreover, it is not considered the recommended framework within FIGIS. See the Web Framework analysis results.
2. Avoid relying on complex ESB and Messaging Systems solutions. ServiceMixand Camelare great enterprise integration platforms but not really in the scope of this exercise.
3. Avoid overriding intersection results. The result of an intersection should be maintained until the same intersection is requested to be recalculated, it should not be overwritten by the next intersection calculation batch.

Functional Requirements

Base Layers

- update to Geoserver 2.1 for equal area projection support (Eckert IV)
- need to provide FI Geoserver with layers well geo-coded:
- For instance:
  e.g. 1: FAO_MAJOR_AREA: well geo-coded with a single code F_AREA for each geo-feature
  e.g. 2: SPECIES_DIST: not well geo-coded: the geocode is a simple integer field.

- provide single geo-features by unique combination of attributes (in this case: ALPHACODE, DISPORDER, PRESENCE).
    - to improve the geo-coding system (here for SPECIES DIST an integer may not be appropriate), would based on a combination (concatenation) of the attributes involved in defining the geo-features. On the contrary: a multiple attributes selection may be possible instead of a unique SRC_CODE (see Geospatial Query Engine p. 10, 14) but it would imply a modification of the STATISTICAL TABLE SCHEMA.

Algorithm / Intersection

- Add projection process in the algorithm using an Equal Area projection (for area calculation purpose), since layers stored in Geoserver are in WGS84 CRS.

  The alternative would be to project all layers in Geoserver in an global equal area projection, in this case, checks regarding on the fly reprojections would be necessary on map viewer applications such as the RFB map viewer.
- A Difference/Erase process would apply similarly as in the current algorithm, to “remove” eventual land overlay
- Accurate Area calculation (based on Equal Area projection)
- Intersection process:
    - multiple attributes selection (multiple columns) if the source layer geo-code is not explicit
    - multiple attribute values selection (e.g. for FAO Major areas : a list of 27,47,31 fao major areas)
- Guarantee the uniqueness of each combination: Each attributes relationship may correspond to a unique geo-feature. This may implies using an additional geometry Union process.
- Discuss the terminology used until now regarding SOURCE and TARGET: An intersection process does not have any order in selecting the base layers. The use of SOURCE and TARGET definitions may not be appropriated and may be confusing since both layers used in the intersection are SOURCE layers. Using a layer 1 as source and a layer 2 as target, or the contrary, would lead to the same combination result.
- Precisions are needed regarding:
    - the database storage of intersection geometries (and possible errors as mentioned in the technical documentation)
    - the support (or not) of layers in which polygons are not exclusive and can overlap between each others (topology not respected).
    - the possibility for the Download manager to support other output Formats than CSV, XML, GML2

Use Cases

1. Feed the water area intersection table
2. Calculate the intersection between two (layer, area) tuples via (REST) web service
3. Scenarios for map viewers:
    a. Species Map Viewer: end-user request one or more species distribution areas and the application returns a list of all interstecting layers (i.e., RFBs, EEZs, High-seas, LMEs, Country (in case of Inland Species), FAO areas) and the intersection area percentage.
    b. Regional Fisheries Bodies Map Viewer: end-user request one RFB area of

competence and the application returns a list of all interstecting layers (i.e., species, other RFBs, EEZs, High-seas, LMEs, Country (in case of Inland RFB), FAO areas) and the intersection area percentage.

c. Each intersecting layer is displayed in the map at user request and the result of the computed intersections is exportable in printable or re-usable format (e.g., PDF, CSV).

[09 June 2011 - 16:48] Meeting Report and proposed solution for first phase

NOTES: It is worth to point out that the report here below is still a draft since some requisites, especially on client side, need to be better defined. Also the proposed solution envisages a first release of Intersection Engine which aimed to be the base for further development as new requirements will be arised or the old one will be changed somehow or better specified.

Examining in details the requirements and use cases with Francesco, Fabio, Aureliano and Alessio, we established for this first phase as follows:

1. The Intersection Engine will get as input an XML conifguration file listing all the possible couple of Sources and Targets. Each line should contain the following information:

   SRC_SYSTEM / TARGET_SYSTEM / SRC_CODE_FIELD / TARGET_CODE_FIELD / PRESERVE_SRC_GEOM_OR_NOT

   e.g.: SPECIES / FAO_DIV / ALPHACODE / F_AREA / FALSE

   the user can specify many configuration. The IE won't take into account equal rows anyway.

2. For each configuration will be generated two different intersecting tables:
   - STATISTIC_TABLE [IE_SRC_TRG_SRCCODE_TRGCODE] --> feature_id / src_system / trg_system / src_code_field_name / src_code_field_value / trg_code_field_name / trg_code_field_value / src_area / trg_area / src_overaly_percentage / trg_overlay_percentage

   - GEOMETRY_TABLE [IE_SRC_TRG_SRCCODE_TRGCODE_GEOM] --> feature_id / geometry

NOTE: *we are still evaluating if it would be better having only one table for all statistics or create a new one for each couple.*

3. We envisage the use of complex code_fields for layers in order to better geo-code the outcomes. For instance another SPECIES_EXT layer will be created containing a code_field defined as the unione of 3ALPHA_PRESENCE_DISPORDER

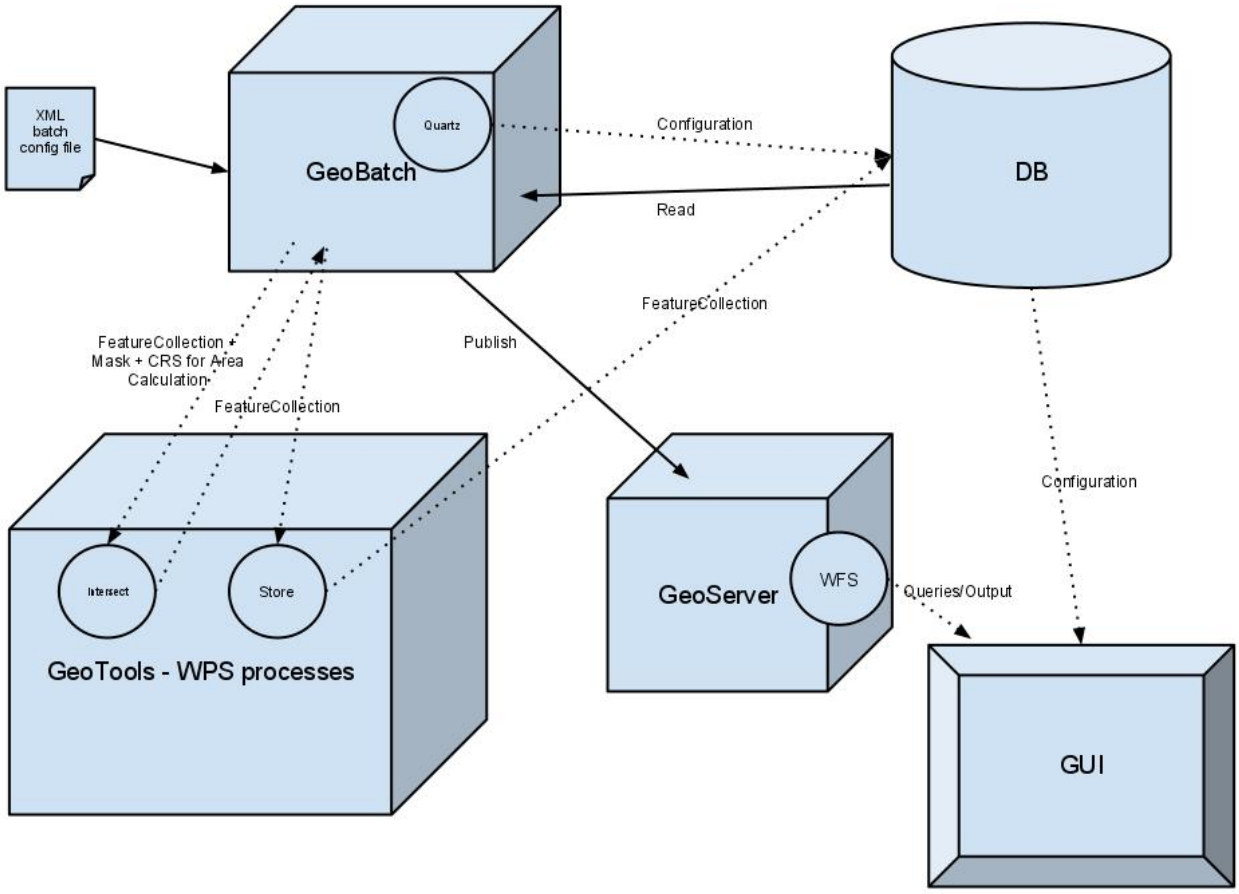
   e.g.: ALB_0_0 can be a possible value of this "composite" field

4. A cron task will launch periodically an Intersection Engine trigger which will build the intersecting tables and layers on GeoServer following the specified algorythm.

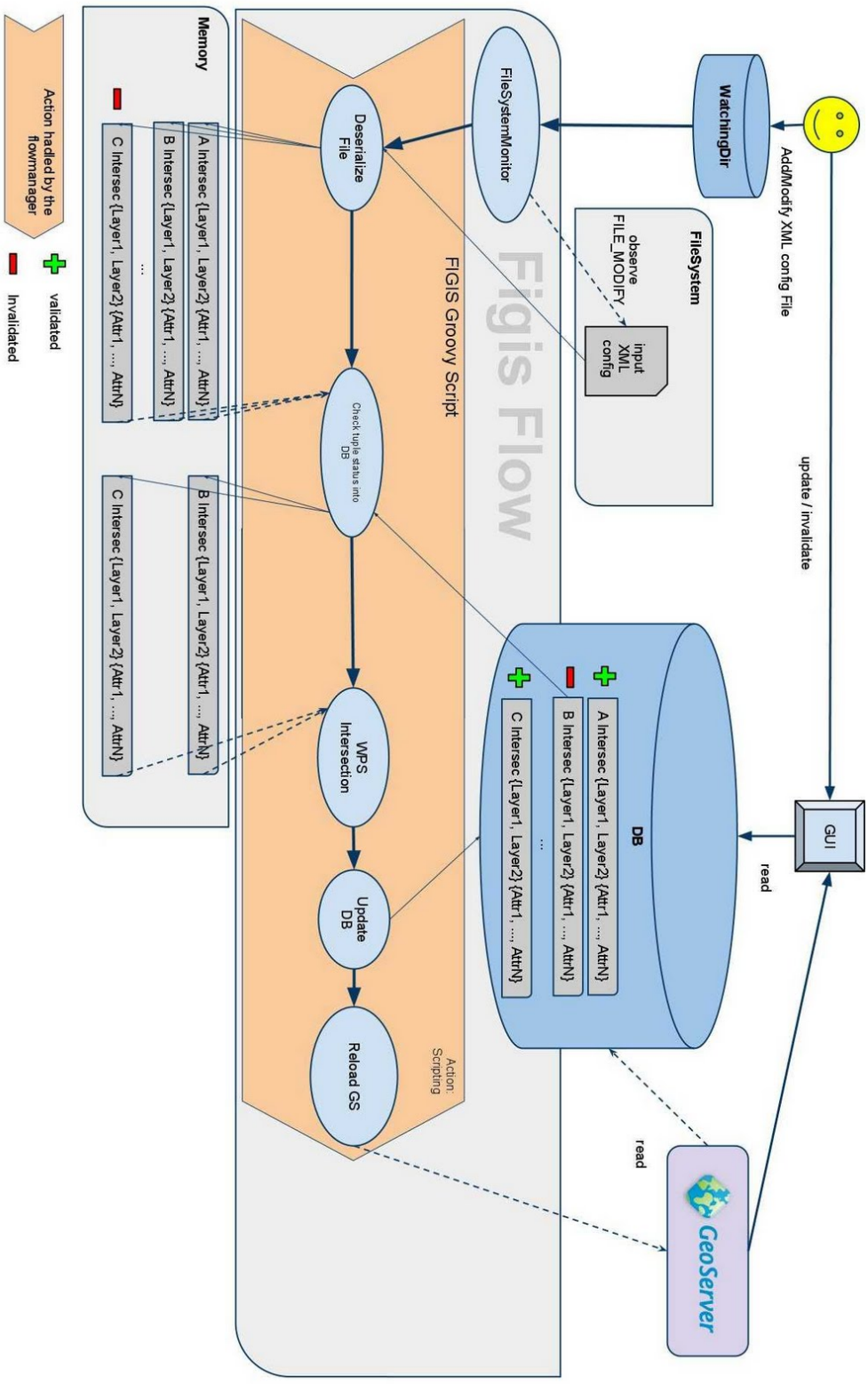
5. The Intersection Engine will allow users to invalidate generated intersecting tables which will be rebuilt. Otherwise the outcomes will be maintained by the IE trigger and won't be generated again.

6. The outputs will be provided by the WFS in different output formats: XML, CSV, GML2, RDF, SHP ... others may be defined later.

Intersection Engine Architecture



Intersection Engine GeoBatch Flow details



Intersection Engine GeoTools processes

Intersection process

This is the reusable process to compute intersections and areas

Inputs: maskCollection, featureCollection1, featureCollection2, areaCRS
Output: a new feature collection whose structure is att1, att2, the intersected geometry, src area, target area, src_overlay_perc, target_ovelay_perc

The mask collection is optional. GeoBatch is supposed to get the two collections from the database directly, already shaving off all attributes but the two that are requested in its configuration.
The two input feature collections are supposed to be in the same CRS too.
The areaCRS is a CRS that will be used to compute the areas of the various polygons generated by the process

Main computation “loop” (it will be done in a feature collection iterator actually, to avoid storing everything in memory):

- get the mask bbox, the fc1 and fc2 bboxes, intersect them all -> that’s the working area
- for each feature in fc1 inside the working area
    - get the corresponding mask features, union their geometry, get the clipping geometry as a result
    - clip the geometry of fc1
    - reproject the geometry in the areaCRS and compute the area
    - get all features in fc2 that can intersect the clipped fc1 geometry
    - for each of the fc2 geometries
        - clip the fc2 geometry to the mask
        - reproject the result in areaCRS to compute the original area
        - intersect with fc1 (this is the result geometry) and clean up eventual point and line elements (they will happen if we intersects geometries that do touch in some parts without actually intersecting)
        - reproject the result in areaCRS to compute the intersected area
        - compute percentages
        - emit the resulting feature with all the non geometric attributes of the first and second feature, the intersected geometry, the src, target area and the overlap percentages

Store process

This one is FIGIS specific
Inputs: statsFeatureCollection, sourceSystem, targetSystem, oracleStore, statsTableName, geomTableName
Outputs: none, will just give back a return status (since we can’t have a process with really 0 ouptuts)

The oraclestore is a geotools JDBCDataStore that GeoBatch already used to grab the feature

collections, we'll get the connection to do the inserts from its connection pool (btw, geobatch could keep the connection pool or the store itself around).

The process will first create two temporary target tables deriving details from the feature collection (the attribute names and types) and the source and target system, then scroll through the collection and perfom batches of inserts in the db (a transaction every, say, 100-1000 records to avoid overwhelming the db, then commit and restart the transaction).
Once done it will remove the eventual old tables and rename the temp ones to their definitive names (in a single transaction)

It is the responsibiliy of GeoBatch to configure the resulting sql view in GeoServer for publishing purposes.