International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 3 (2011) pp.119-126
© MIR Labs, www.mirlabs.net/ijcisim/index.html

The SQL-Based Geospatial Web Processing Service

Soravis Supavetch and Sanphet Chunithipaisan

Geo-Image Technology Research Unit, Department of Survey Engineering,
Faculty of Engineering, Chulalongkorn University, Bangkok, Thailand
p_soravis@hotmail.com, sanphet.c@chula.ac.th

***Abstract*: Since the OGC promoted the specification of OGC Web Processing Service (WPS), several researches on the development of geospatial online service have been happened. Most of WPSs are developed extending on GIS software or APIs. Nowadays, most databases are extended to support spatial data and also provide a number of spatial functions which can be called through SQL. This paper presents the development of WPS which uses spatial database as processing engine. Extended WPS protocol is designed to support SQL to invoke spatial function. Using SQL eases the modification of WPS regarding the change of processing function service. Databases tested with the implemented WPS include Oracle, PostgreSQL and MySQL. The database type is listed in WPS GetCapabilites document and allowed to choose for working spatial engine. Thus, users can use SQL in which they are familiar with the database. The results from scenario tests clearly show the success of the implementation of this concept.**

***Keywords*:** Web Processing Service, Geospatial Processing Service, Geospatial Processing Engine, Spatial Database.

I. Introduction

Traditional GIS systems are no longer suitable for modern distributed, heterogeneous network environments due to the lack of interoperability, reusability, and flexibility [2]. The geographic information service is enabled by the advancement in general web service technology and the efforts of the GIS industries. Due to the popular use of the Internet and the dramatic progress of telecommunication technology, the paradigm of GIS is shifting into a new direction [6], especially for producing spatial-oriented visualization [5] in personal devices such as cell phones, PDAs, and smart phones [4]. Until the Open Geospatial Consortium (OGC), the non-profit, international, voluntary consensus standards organization is leading the development of standards for geospatial and location based services. Several implementation specifications have been deployed, and some of them are widely accepted in GIS industries e.g. GML, WMS, WFS, and WPS. These services open the opportunity to solve the problems that are typically faced in traditional GIS. The OGC WFS [8] defines interfaces for data access and manipulation operations on geographic features. The OGC WMS [10] specifies the behavior of a service that produces image maps. The OGC WPS [11] defines a mechanism by which a client may submit a processing task to server. To implement these OWSs supporting scientific workflow require services composition called “web services chaining” [7], “web services composition”, “web services flow” or “web services orchestration” [12].

Even though WPS fulfils the gap of transition from old traditional standalone GIS through service-based architecture, no real geo-processing capabilities do exist. For reasoning of ignoring re-implementation an existing GIS function, [1] developed WPS using GRASS GIS for a processing engine and open shell script through WPS interface to user for executing GRASS functions. On our perspective, not only GIS software can be used as WPS back-end processing engine, but also spatial database is suitable. Spatial database nowadays provides numerous spatial functions. In addition, the spatial functions can be customized or extended through a stored procedure.

Currently, several relational databases provide spatial engine to perform several spatial operations which can be used with stored data through SQL without the need of external GIS tools. If spatial database can be used as the spatial engine for the back-end processing of WPS, this will help reduce workloads for programmer to develop spatial operation tools. Moreover, if SQL statement can be attached in the request form, the translation of XML in the request form invoking the GIS functions can be dismissed since it is directly sent to spatial database for carrying out the GIS function. It is also flexible for users to form SQL with knowledge on using back-end spatial database used in WPS.

This paper reports the implementation of WPS using the spatial database as the back-end spatial engine for processing wfs:FeatureCollections data from open WFS site. This research is an adaptation of our previous research [3] in which WPS request form is extended to support the additional basic SQL (SELECT-FROM-WHERE-FILTER) for calling the processing task from available databases in WPS. Oracle Spatial, PostGIS and MySQL are databases that are used in this research.

Next section gives a short insight into WFS, WPS and Spatial Database. The detail of extended spatial SQL for WPS request is presented. The developed WPS architecture and back-end procedural composition are introduced. The processing capabilities of developed system are described. The test of implemented service through some scenarios is demonstrated.

II. Technologies Reviews

A. WFS (Web Feature Service)

WFS enables a client to retrieve geospatial data encoded in Geography Markup Language (GML) online through HTTP protocol. It allows for clients and servers of different vendors or systems to share data without having to convert data between proprietary formats. WFS server is then a feature online resource, providing an interface for data access and manipulation. A request sent to a WFS server is a query (or transformation operation) for one or more features that can be delivered by a server. When the WFS receives the request, it executes the function by the request and sends back the information to a client. WFS communicates with a client and allow programs written in different languages and on different platforms. Interface that is typically used in WFS is defined in XML.

B. *WPS (Web Processing Service)*

WPS is designed to standardize the way that GIS calculations are made available to the Internet. WPS can describe any calculation (i.e. process) including all of its inputs and outputs, and triggers its execution as a Web service. WPS supports simultaneous exposure of processes via HTTP/GET (KVP: Key Value Pairs) and HTTP/POST (XML-based document), thus allowing the client to choose the most appropriate interface mechanism. The specific processes served up by a WPS implementation are defined by the owner of that implementation.

C. *Spatial Database*

Spatial database is a database system which offers spatial data types in its data model and query language. Most relational database management systems have supported spatial data type including point, line and polygon. Some of them include Oracle, MySQL and PostgreSQL. MySQL has included spatial extension built in since MySQL 4.1 which supports data type following OGC simple feature specifications for SQL, and also provided spatial query tools and supports spatial indexing using R-tree with quadratic splitting. PostGIS is used to spatially enable the PostgreSQL to be used as a backend spatial database. PostGIS uses an R-tree index implemented on top of GiST (Generalized Search Trees) to index GIS data. Oracle Spatial is a module option for Oracle that provides advanced spatial features to support GIS solutions. OGC simple feature is supported since Oracle 10g. An R-tree or quad tree index can be used in Oracle for spatial data index.

III. Extend Spatial SQL for WPS

SQL as processing command is the main idea of this research to explore processing capabilities of spatial database to web-based user. An SQL base syntax SELECT-FROM-WHERE clause is applied to SELECT-FROM-WHERE-FILTER for WPS user request. Extended SQL is designed for user to send a request statement to WPS which enable user to define a GetFeature from WFS site with processing simultaneously. Processing functions are available in our WPS coming from spatial database. The pattern of designed SQL statement is shown in the following section.

A. *Processing Clause*

Only SELECT clause from database SQL is un-extended. User can use spatial functions e.g., SDO_CS, SDO_GEOM, SDO_LRS or SDO_UTIL package subprograms from Oracle, ST_Buffer, ST_DWithin, ST_Transform, ST_Line_Interpolate_Point functions from PostGIS, or functions from MySQL, to process features from WFS site.

Another advantage of using spatial database as processing engine is that user can customize additional function embedded to database through stored procedure. More developed algorithms can be plugged into service. The processing function in SELECT clause is not limited in one function per request but user can request numerous examples. For example SELECT ST_Buffer(ST_Transform(…) , 100), ST_Buffer(ST_Transform(…) , 200) to retrieve two different buffered polygons from single original polygon within coordinates already transformed.

B. *Data Retrieving Clause*

This processing service provides only processing functions without any provided features. Processing features must retrieve from WFS site. We extended an SQL in FROM clause to express WFS site and attaching FILTER clause for defining filter condition. The address of WFS site is located in the FROM clause. The layer name is defined after the address separated by #. The address and the layer name are both encoded together inside <?--?> tag. For filtering condition, user can express OGC Filter Encoding like SQL. We use CQL filter (Common Query Language) from GeoTools (http://www.geotools.org) to wrap FE (Filter Encoding) [9] expression. The following table is examples of CQL filter comparing with OGC FE syntax.

CQL	Filter Encoding
L.NAME = ‘LAND01’	<PropertyIsEqualsTo/>
L.NAME <> ‘LAND01’	<PropertyIsNotEqualsTo/>
L.AREA > 100	<PropertyIsGreaterThan/>
BBOX(GEOM, 10,20,30,40)	<BBOX />
CONTAINS(GEOM, …)	<Contains/>
INTERSECT(GEOM, …)	<Intersect/>

Table 1. A Comparison of CQL Filter and OGC Filter Encoding

Syntax Rules

INSERT, UPDATE and DELETE clauses are not allowed for this service. User only use SELECT clause to request a processing function. The structure of extended SQL statement is as follows:

```
SELECT  <Processing Function>
FROM    <WFS Site>
WHERE   <Processing Condition>
FILTER  <CQL Filter>
```

An expression in the SELECT clause is a native SQL statement according to selected spatial database by user. FROM clause is defined for specifying WFS URL and the name of feature type. WHERE clause is an option for defining a processing condition. The filter encoding which is used in diverted WFS request is after the FILTER clause. The following statement is used to demonstrate the sample of requested SQL to different processing engine.

```
SELECT  L.NAME,
        SDO_GEOM.RELATE(L.SHAPE,'determine',
        SDO_UTIL.FROM_WKTGEOMETRY(
        'POLYGON((
        101.86 14.81,101.86 14.92,101.97 14.92
        ,101.86 14.81))'),0.1)  AS RELATIONSHIP
FROM   <?http://localhost/geoserver/wfs#nesdb:LAND?>
FILTER  NAME = 'LAND01'

SELECT  L.NAME,
        MBROverlaps(L.SHAPE,'determine',
        GeomFromText('POLYGON((
        101.86 14.81,101.86 14.92,101.97 14.92
        ,101.86 14.81))'),0.1) AS  RELATIONSHIP
FROM <?http://localhost/geoserver/wfs#nesdb:LAND?>
FILTER  NAME = 'LAND01'
```

The first example is to evaluate the polygon relationship between the land parcel 'Land01' which is queried from another WFS and the defined polygon. Such processing occurrs in Oracle database. The second example is the same question as the first example, but the spatial engine used is MySQL database.

IV. A WPS Archetecture

The developed WPS used existing spatial database, Oracle Spatial, MySQL and PostgreSQL, as a spatial engine. Several tools were developed for many purposes, e.g., to support extended SQL request form, to connect and retrieve data from other WFS sites and to manage and manipulate collected GML data. The architecture of the service system is shown in Figure 1.

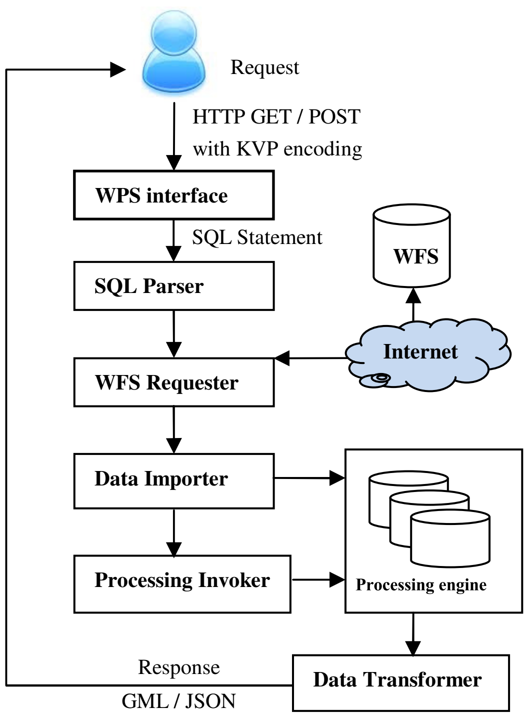


Figure 1. Online processing service frame work.

When the user send the request via HTTP GET or POST with KVP encoding which is a designed statement including SQL for processing function, SQL parser will interpret and extract a WFS data source and SQL statement from the user request to form a WFS request. WFS requester will send such request to WFS site to get data in GML format. Once GML stream is responded back from WFS site, the data importer will transfer that received data into the database store. After that, the processing invoker will send the SQL command to the spatial database to call spatial processing functions. The result set from spatial database will be parsed to a data transformer for converting result set to user defined output format: GML or JSON (JavaScript Object Notation).

The sequence diagram of above processing tasks is demonstrated as follows:

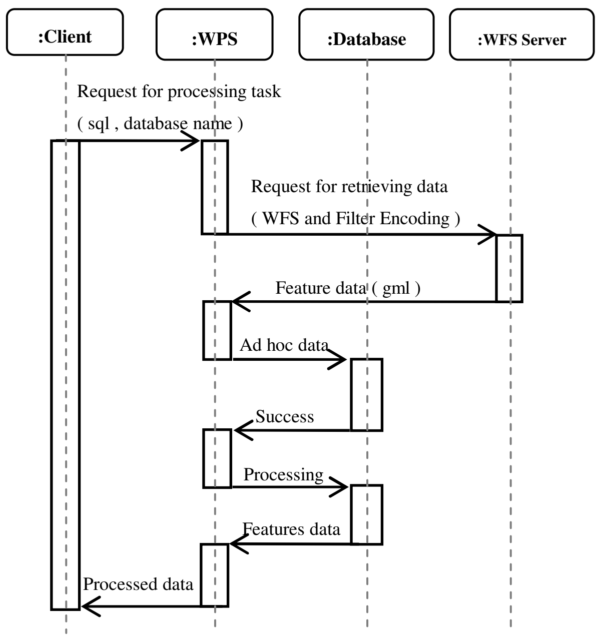


Figure 2. Online query and processing service (OQPS) sequence diagram

A. SQL Parser

An SQL Parser component is the most importance part of the service. Every incoming SQL composes of two conditions i.e., data retrieving condition and data processing condition. Data requests from FORM and FILTER clause will be re-encoded in WFS request protocol in order to parse to WFS data source. Data that are retrieved from WFS site will be stored in ad hoc database table. The user requests statement is then reformed into database native SQL for sending to user defined database to invoke spatial operation.

Requested SQL

```
SELECT    <Processing Functions>
FROM      <URL#TypeName>
WHERE     <Processing Condition>
FILTER    <CQL Filter>
```

Executing SQL

```
SELECT    <Processing Functions>
FROM      <Temporary table>
WHERE     <Processing Condition>
```

Figure 3. Recomposing requested SQL.

B. WFS Requester

WFS Requester is to establish the connection to other WFS site for sending a request and retrieving a response in GML. Before importing retrieved features to database, WFS Requester establishes the connection through GetCapabilities and GetDescribeFeatureType for evaluating feature schemas and spatial reference system for spatial database table preparation. Information from GetCapabilities and GetDescriveFeatureType are necessary for creating metadata of feature table in spatial database. The working sequence of WFS Requester with other related components of the system is shown in Figure 4.

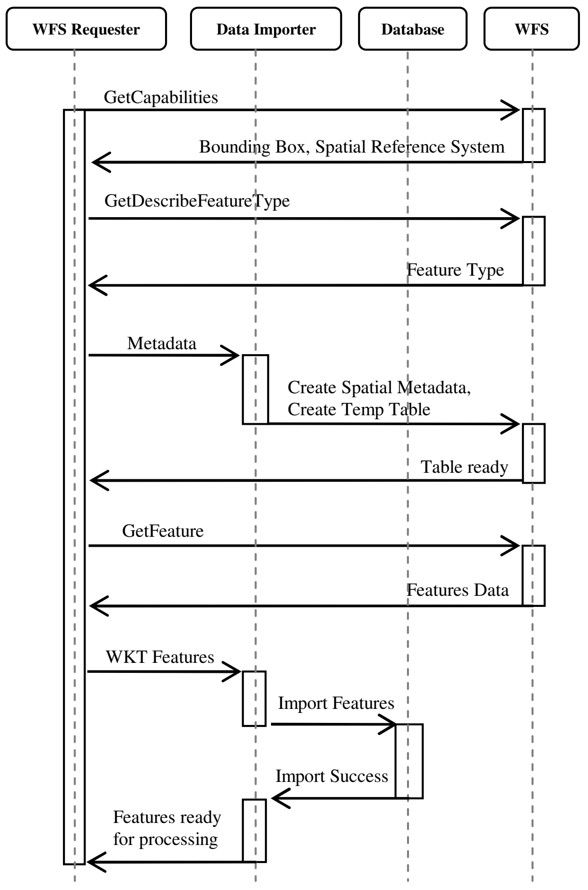


Figure 4. Preparation diagram for temporal feature importation.

C. Data Importer

Data Importer imports data from WFS requester into prepared table for ad-hoc dataset to be used in the processing task later. Data Importer transforms GML feature members into SQL insert statement and performs execution. For using Oracle database, after data inserted into table, Data Importer will invoke creating spatial index statement (Oracle also requires spatial index for performing spatial query and processing).

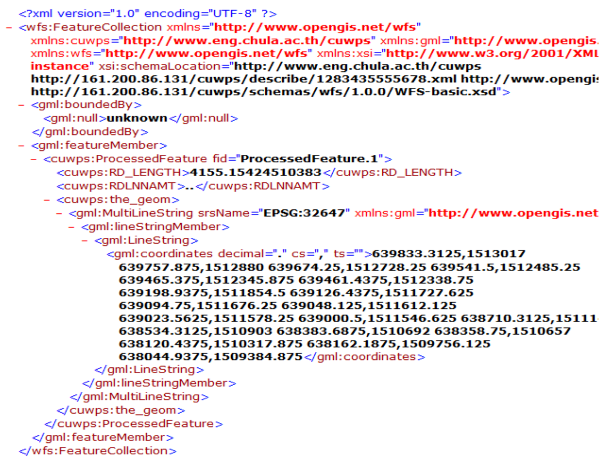
D. Processing Invoker

Processing Invoker will send a recomposed SQL statement from SQL Parser to database for carrying out spatial processing operation.

E. Data Transformer

The resulting set from the output of spatial database will be transformed in GML or JSON (upon the user request) which is done by using the Data Transformer tool. The features attributes result is identified and extracted from the user request statement in SELECT clause. The attribute which comes from the processing operation is defined using alias name (defined using AS …). The following example demonstrates the GML result from database execution according to a user request statement.

```
SELECT    SDO_GEOM.SDO_LENGTH(
          SDO_AGGR_CONCAT_LINES(THE_GEOM),0.05)
          AS RD_LENGTH,
          RDLNNAMT,
          SDO_UTIL.TO_GMLGEOMETRY(
          SDO_AGGR_CONCAT_LINES(THE_GEOM))
          AS THE_GEOM
FROM   <?http://161.200.86.131/geoserver/wfs#cuwps:mainroad?>
GROUP BY RDLNNAMT
FILTER    MAINROAD_I = 78230
```



```
  <?xml version="1.0" encoding="UTF-8" ?>
- <wfs:FeatureCollection xmlns="http://www.opengis.net/wfs"
    xmlns:cuwps="http://www.eng.chula.ac.th/cuwps" xmlns:gml="http://www.opengis.
    xmlns:wfs="http://www.opengis.net/wfs" xmlns:xsi="http://www.w3.org/2001/XMI
    instance" xsi:schemaLocation="http://www.eng.chula.ac.th/cuwps
    http://161.200.86.131/cuwps/describe/1283435555678.xml http://www.opengi
    http://161.200.86.131/cuwps/schemas/wfs/1.0.0/WFS-basic.xsd">
  - <gml:boundedBy>
      <gml:null>unknown</gml:null>
    </gml:boundedBy>
  - <gml:featureMember>
    - <cuwps:ProcessedFeature fid="ProcessedFeature.1">
        <cuwps:RD_LENGTH>4155.15424510383</cuwps:RD_LENGTH>
        <cuwps:RDLNNAMT>..</cuwps:RDLNNAMT>
      - <cuwps:the_geom>
        - <gml:MultiLineString srsName="EPSG:32647" xmlns:gml="http://www.opengis.net
          - <gml:lineStringMember>
            - <gml:LineString>
                <gml:coordinates decimal="." cs="," ts="">639833.3125,1513017
                  639757.875,1512880 639674.25,1512728.25 639541.5,1512485.25
                  639465.375,1512345.875 639461.4375,1512338.75
                  639198.9375,1511854.5 639126.4375,1511727.625
                  639094.75,1511676.25 639048.125,1511612.125
                  639023.5625,1511578.25 639000.5,1511546.625 638710.3125,15111
                  638534.3125,1510903 638383.6875,1510692 638358.75,1510657
                  638120.4375,1510317.875 638162.1875,1509756.125
                  638044.9375,1509384.875</gml:coordinates>
              </gml:LineString>
            </gml:lineStringMember>
          </gml:MultiLineString>
        </cuwps:the_geom>
      </cuwps:ProcessedFeature>
    </gml:featureMember>
  </wfs:FeatureCollection>
```

Figure 5. GML result.

From the above example, RD_LENGTH and THE_GEOM are used as alias names of feature attributes with the attribute data coming from the processing operations, whereas, RDLNNAMT is the attribute which already come from the requested WFS and can be directly put in GML.

Another format, JSON, is also supported in the Data Transformer tool. This type of output can be directly used in client side application which JavaScript is supported, e.g. OpenLayers and GeoExt. JSON data is encoded in the format so-called JavaScript array object which is used to store multiple values in a single variable. Some JavaScript library can benefits from JSON format for data visualization by rendering data in table form, for example, Ext.data.ArrayStore class in Ext. The following example shows the output in JSON format from the same user request statement.

```
SELECT  SDO_GEOM.SDO_LENGTH(…)
        AS RD_LENGTH,
        RDLNNAMT,
        SDO_UTIL.TO_WKTGEOMETRY(
        SDO_AGGR_CONCAT_LINES(…) )
        AS THE_GEOM
FROM    <?...?>
FILTER    …
```

```
[<string: RD_LENGTH>,<string:RDLNNAMT>,<string:WKT>]
```

Figure 6. Additional JSON output format.

V. Processing Capabilities

The implemented service information and detail is available at http://161.200.86.131/cuwps/wps.jsp which provides a link to access GetCapabilities and DescribeProcess operations.

The processing functions available in the system can be checked through GetCapabilities operation. There is only one type of processing function available in the system, called SQL_PROCESSING, since the spatial operations can be freely defined by user in SQL statements depending on which database is used. The following example demonstrates the GetCapabilities document which is encoded in XML.

```
<ProcessDescription processVersion="1" … >
   <cuwps:Identifier>SQL_PROCESSING
   </cuwps:Identifier>
   <cuwps:Title>
   SQL expression for a processing task …
   </cuwps:Title>
   …
</ProcessDescription>
```

User can use any SQL statement in which database that the user is familiar with. The detail of using the SQL_Processing service is found at DescribeProcess operation. It describes the parameters that have to be sent to the service, the type and value of parameters to be defined, and the output type of response value. The example of DescribeProcess document of SQL_Processing is shown as follows:

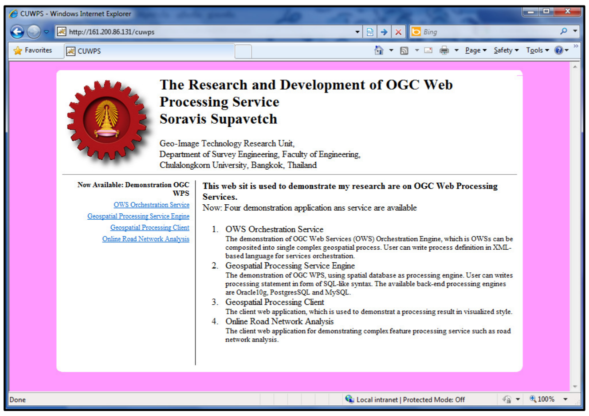
```
<DataInputs>
 <Input minOccurs="1" maxOccurs="1">
   <cuwps:Identifier>SQL</cuwps:Identifier>
   <cuwps:Title>SQL statement</cuwps:Title>
   <cuwps:Abstract>SQL statement
   </cuwps:Abstract>
   <LiteralData>
      <cuwps:DataType ows:reference="xs:string"/>
      <cuwps:AllowedValues>
      <cuwps:Value/>
      </cuwps:AllowedValues>
   </LiteralData>
</Input>
<Input minOccurs="1" maxOccurs="1">
   <cuwps:Identifier>DATABASE</cuwps:Identifier>
   <cuwps:Title>Database name</cuwps:Title>
   <cuwps:Abstract>Database name
   </cuwps:Abstract>
      <LiteralData>
         <cuwps:DataType ows:reference="xs:string"/>
         <cuwps:AllowedValues>
         <cuwps:Value>ORACLE_10G</cuwps:Value>
         <cuwps:Value>MYSQL_5</cuwps:Value>
         <cuwps:Value>POSTGIS_8_4</cuwps:Value>
         </cuwps:AllowedValues>
      </LiteralData>
</Input>
</DataInputs>
<ProcessOutputs>
   <Output>
      <cuwps:Identifier>GML2</cuwps:Identifier>
```

```
        <cuwps:Title>GML document version
        2.0</cuwps:Title>
        <cuwps:Abstract>GML document from processing
        features </cuwps:Abstract>
    </Output>
    <Output>
        <cuwps:Identifier>JSON</cuwps:Identifier>
        <cuwps:Title>Array of processing
        result</cuwps:Title>
        <cuwps:Abstract>Javascript
        arrays</cuwps:Abstract>
    </Output>
    <ComplexOutput>
        <Default>
            <Format>
            <MimeType>text/xml </MimeType>
            <Schema>http://161.200.86.131/cuwps/
            schemas/gml/2.1.2/gml.xsd</schema>
    …</ProcessOutputs>
```

As shown above, user has to define the type of database to be used for processing function by defining the value for the DATABASE identifier: ORACLE_10G, MYSQL_5 or POSTGIS_8_4. User put the SQL statement that matches the SQL with defined database and follows the designed SQL syntax. There are two types of output that can be chosen including GML2 and JSON. The default output type is GML2.

VI. Service Testing and Sample Application

We developed demo client application through our web site (http://161.200.86.131/cuwps/demo_wps.jsp) for showing facilities of using extended SQL from spatial database for processing features from WFS site by CUWPS (Chulalongkorn University Web Processing Service).


CUWPS - Windows Internet Explorer
http://161.200.86.131/cuwps
Bing
Favorites
CUWPS
Page
Safety
Tools
Done
Local intranet | Protected Mode: Off
100%

The Research and Development of OGC Web Processing Service
Soravis Supavetch

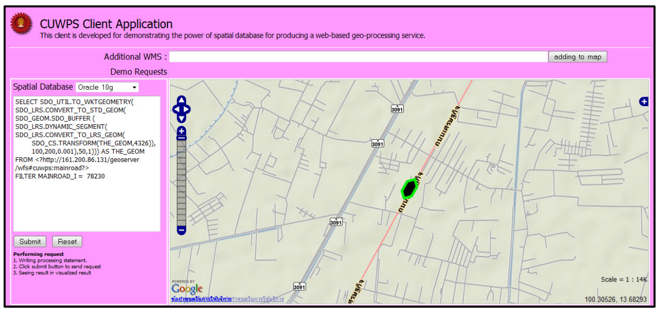
Geo-Image Technology Research Unit,
Department of Survey Engineering, Faculty of Engineering,
Chulalongkorn University, Bangkok, Thailand

Now Available: Demonstration OGC WPS
OWS Orchestration Service
Geospatial Processing Service Engine
Geospatial Processing Client
Online Road Network Analysis

This web sit is used to demonstrate my research are on OGC Web Processing Services.
Now: Four demonstration application ans service are available

1. OWS Orchestration Service
   The demonstration of OGC Web Services (OWS) Orchestration Engine, which is OWSs can be composited into single complex geospatial process. User can write process definition in XML-based language for services orchestration.
2. Geospatial Processing Service Engine
   The demonstration of OGC WPS, using spatial database as processing engine. User can writes processing statement in form of SQL-like syntax. The available back-end processing engines are Oracle10g, PostgresSQL and MySQL.
3. Geospatial Processing Client
   The client web application, which is used to demonstrat a processing result in visualized style.
4. Online Road Network Analysis
   The client web application for demonstrating complex feature processing service such as road network analysis.

Figure 7. CUWPS web site (http://161.200.86.131/cuwps/).



```
SELECT SDO_UTIL.TO_WKTGEOMETRY(
SDO_LRS.CONVERT_TO_STD_GEOM(
SDO_GEOM.SDO_BUFFER (
SDO_LRS.DYNAMIC_SEGMENT(
SDO_LRS.CONVERT_TO_LRS_GEOM(
      SDO_CS.TRANSFORM(THE_GEOM,4326)),
      100,200,0.001),50,1))) AS THE_GEOM
FROM <?http://161.200.86.131/geoserver
/wfs#cuwps:mainroad?>
FILTER MAINROAD_I =  78230
```

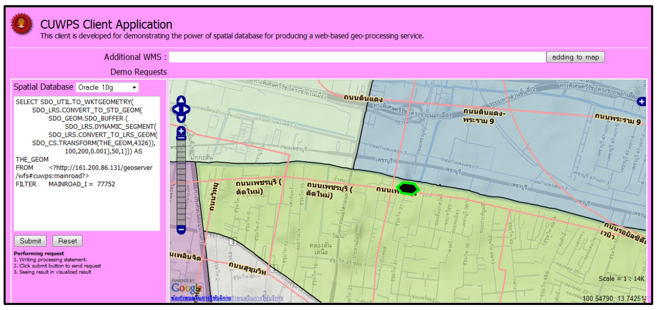
Performing request
1. Writing processing statement.
2. Click submit button to send request
3. Seeing result in visualized result

Figure 8. Demo client application using OpenLayers to visualize result (http://161.200.86.131/cuwps/demo_wps.jsp).

Some scenarios were set up in order to test the concept of this implementation and the developed system. The WFS was set up using GeoServer with the data of Bangkok Metropolitan Administration (BMA) including road and district layers.

The first scenario is to demonstrate the use of Linear Referencing System (LRS) function available from Oracle Spatial to generate a 50-meter buffer around the road segment from 100 meters to 200 meters from the starting point of MAINROAD_I = 77752. The encoding for the STATEMENT and visualized result are shown as follows respectively:

```
SELECT  SDO_UTIL.TO_WKTGEOMETRY(
        SDO_LRS.CONVERT_TO_STD_GEOM(
        SDO_GEOM.SDO_BUFFER (
            SDO_LRS.DYNAMIC_SEGMENT(
            SDO_LRS.CONVERT_TO_LRS_GEOM(
            SDO_CS.TRANSFORM(THE_GEOM,4326)),
            100,200,0.001),50,1))) AS THE_GEOM
FROM <?http://161.200.86.131/geoserver/wfs
        #cuwps:mainroad?>
FILTER  MAINROAD_I =  77752
```

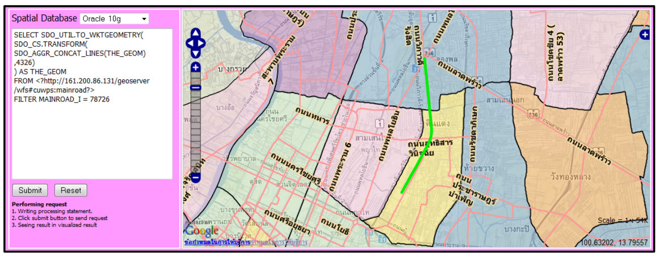


```
SELECT SDO_UTIL.TO_WKTGEOMETRY(
      SDO_LRS.CONVERT_TO_STD_GEOM(
            SDO_GEOM.SDO_BUFFER (
                  SDO_LRS.DYNAMIC_SEGMENT(
            SDO_LRS.CONVERT_TO_LRS_GEOM(
      SDO_CS.TRANSFORM(THE_GEOM,4326)),
                  100,200,0.001),50,1))) AS
THE_GEOM
FROM       <?http://161.200.86.131/geoserver
/wfs#cuwps:mainroad?>
FILTER     MAINROAD_I =  77752
```

Performing request
1. Writing processing statement.
2. Click submit button to send request
3. Seeing result in visualized result

Figure 9. Buffered polygon returned from CUWPS by executing Oracle LRS function.

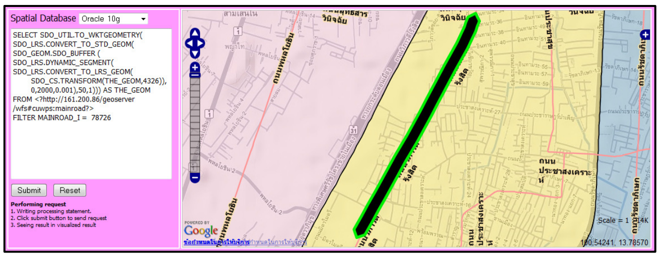
The above example uses the Oracle Spatial for processing service. The requested WFS data will be downloaded and stored in the database. Once the download data is in the database, the processing will be carried out on the ad-hoc data store. The result of the user request will be returned to the user. Figure 10 and 11 shows the downloaded data and data result that are stored in the Oracle Spatial respectively.



```
SELECT SDO_UTIL.TO_WKTGEOMETRY(
SDO_CS.TRANSFORM(
SDO_AGGR_CONCAT_LINES(THE_GEOM)
,4326)
) AS THE_GEOM
FROM <?http://161.200.86.131/geoserver
/wfs#cuwps:mainroad?>
FILTER MAINROAD_I = 78726
```

Performing request
1. Writing processing statement.
2. Click submit button to send request
3. Seeing result in visualized result

Figure 10. VIPAVADEE road data by WFS site.



```
SELECT SDO_UTIL.TO_WKTGEOMETRY(
SDO_LRS.CONVERT_TO_STD_GEOM(
SDO_GEOM.SDO_BUFFER (
SDO_LRS.DYNAMIC_SEGMENT(
SDO_LRS.CONVERT_TO_LRS_GEOM(
        SDO_CS.TRANSFORM(THE_GEOM,4326)),
        0,2000,0.001),50,1))) AS THE_GEOM
FROM <?http://161.200.86/geoserver
/wfs#cuwps:mainroad?>
FILTER MAINROAD_I =  78726
```

Performing request
1. Writing processing statement.
2. Click submit button to send request
3. Seeing result in visualized result

Figure 11. Buffered polygon from CUWPS execution.

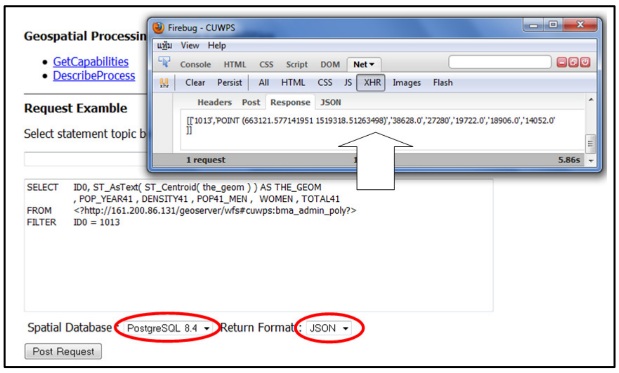
The second scenario is to find the area of the queried polygon from the WFS feature using Oracle Spatial database. The following example shows the SQL statement for finding the area of the object from the WFS feature.

```
SELECT   SDO_UTIL.TO_GMLGEOMETRY(
         THE_GEOM) AS THE_GEOM,
         SDO_GEOM.SDO_AREA(THE_GEOM,10)
         AS DISTRICT_AREA
FROM     <?http://161.200.86.131/geoserver/wfs
         #cuwps:bma_admin_poly?>
FILTER   ID0 = 1013
```

The third scenario demonstrates how to find the centroid of the queried polygon. The following example shows the SQL statement to find the centroid of the district with ID0 1013 from WFS.

```
SELECT   ID, AsText(Centroid(the_geom))
         AS THE_GEOM , POP_YEAR41 ,
         DENSITY41 , POP41_MEN
         ,WOMEN, TOTAL41
FROM     <?http://161.200.86.131/geoserver/wfs
         #cuwps:bma_admin_poly?>
FILTER   ID0 = 1013
```

The above processing expression is carried out using PostgreSQL database, and the result is returned in JSON as shown in Figure 12.



```
[['1013','POINT (663121.577141951 1519318.51263498)','38628.0','27280','19722.0','18906.0','14052.0'
]]
```

```
SELECT   ID0, ST_AsText( ST_Centroid( the_geom ) ) AS THE_GEOM
         , POP_YEAR41 , DENSITY41 , POP41_MEN ,  WOMEN , TOTAL41
FROM     <?http://161.200.86.131/geoserver/wfs#cuwps:bma_admin_poly?>
FILTER   ID0 = 1013
```

Spatial Database : PostgreSQL 8.4 Return Format : JSON

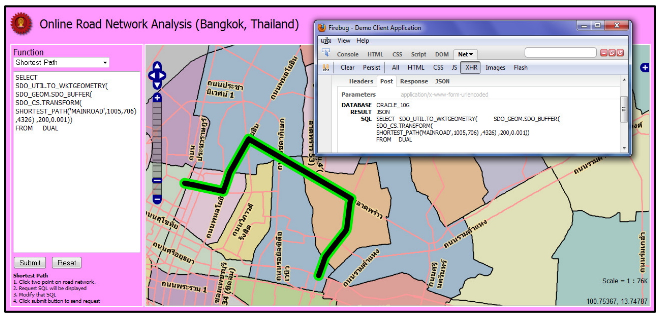
Figure 12. A returned JSON format from CUWPS.

VII. A Complex Feature Processing Service Demonstration Application

To emphasize benefits of using spatial database for WPS implementation, the complex feature model like road network was selected for a demonstration. The network model e.g., node, link and path is complex to exchange over the web for processing through other WPS due to requiring a specific functions for accessing and processing. The PL/SQL sub-programs i.e., SHORTEST_PATH, TSP_PATH (Travel Sale Man Problem) are designed for user request and user can also refer to other WFS site for the data which requires processing with the road network through extended SQL. The Oracle database is selected for use as spatial engine. The Bangkok road network is imported to Oracle and ready to perform a processing by user. The following figures are examples of predefined functions i.e., SHORTEST_PATH and TSP_PATH encoded in request SQL form.

Shortest Path

```
SELECT   SDO_UTIL.TO_WKTGEOMETRY(
         SDO_GEOM.SDO_BUFFER(
          SDO_CS.TRANSFORM(
         SHORTEST_PATH('MAINROAD',1005,706)
          ,4326) ,200,0.001))
FROM     DUAL
```


Online Road Network Analysis (Bangkok, Thailand)
Function
Shortest Path
Submit
Reset
Firebug - Demo Client Application
แฟ้ม View Help
Console HTML CSS Script DOM Net
Clear Persist All HTML CSS JS XHR Images Flash
Headers Post Response JSON
Parameters application/x-www-form-urlencoded
Scale = 1 : 76K
100.75367, 13.74787

```
SELECT
SDO_UTIL.TO_WKTGEOMETRY(
SDO_GEOM.SDO_BUFFER(
SDO_CS.TRANSFORM(
SHORTEST_PATH('MAINROAD',1005,706)
,4326) ,200,0.001))
FROM      DUAL
```

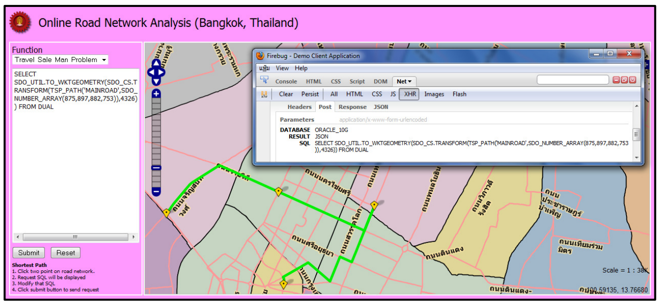
Shortest Path
1. Click two point on road network.
2. Request SQL will be displayed
3. Modify that SQL
4. Click submit button to send request

```
DATABASE  ORACLE_10G
  RESULT  JSON
     SQL  SELECT  SDO_UTIL.TO_WKTGEOMETRY(        SDO_GEOM.SDO_BUFFER(
          SDO_CS.TRANSFORM(
          SHORTEST_PATH('MAINROAD',1005,706) ,4326) ,200,0.001))
          FROM     DUAL
```

Figure 13. Buffered LineString from shortest path.

Travel Sale Man Problem

```
SELECT  SDO_UTIL.TO_WKTGEOMETRY(
        SDO_CS.TRANSFORM(
        TSP_PATH('MAINROAD',
        SDO_NUMBER_ARRAY(875,897,882,753)
        ),4326))
FROM    DUAL
```



```
SELECT
SDO_UTIL.TO_WKTGEOMETRY(SDO_CS.T
RANSFORM(TSP_PATH('MAINROAD',SDO_
NUMBER_ARRAY(875,897,882,753)),4326)
) FROM DUAL
```

Shortest Path
1. Click two point on road network.
2. Request SQL will be displayed
3. Modify that SQL
4. Click submit button to send request

```
Parameters  application/x-www-form-urlencoded
DATABASE  ORACLE_10G
  RESULT  JSON
     SQL  SELECT SDO_UTIL.TO_WKTGEOMETRY(SDO_CS.TRANSFORM(TSP_PATH('MAINROAD',SDO_NUMBER_ARRAY(875,897,882,753
          )),4326)) FROM DUAL
```

Figure 14. TSP_PATH (Travel Sale Man Problem) function visualized result.

VIII. Conclusion

This paper introduces the concept of applying spatial database and using SQL for extending the capabilities of WPS implementation. The data source from WFS site is designed as a parameter in the outlined SQL. The scenario tests show the benefit that user can process geospatial data from other WFS site using common SQL of which database user familiar with.

This WPS framework implements OGC WPS interface without any spatial functions implementation. User can change the execution engine (database) to a new version or others database software. Additionally, this research method can be used to solve a lack of features filtering in WPS (in standard specification, WPS does not necessary provide Filter Encoding) in which the filtering statement can be expressed in WHERE clause, for extracting the processed result.

Thus, all the scenarios demonstrate the benefits of the processing service on data which come from external data source through WFS which using the power of spatial database. The supporting processing service using multiple WFS data sources is the main issue of this research for further work.

Acknowledgments

This research was supported by the Ratchadaphisek Somphot Endownment Grants from Chulalongkorn University.

References

[1] Brauner, J., and Schaeffer, B. “Integration of GRASS Functionality in Web based SDI Service Chains.” Free and Open Source Software for Geospatial conference, Barcelona, Spain, 2008.

[2] Chen, L., Xiujun, M., Guanhua, C., Yanfeng, S., and Xuebing, F. "A Peer-to-Peer Architecture for Dynamic Executing GIS Web Service Composition." *IEEE International Geoscience and Remote Sensing Symposium*, Seoul, Korea (South), 2005.

[3] Chunithipaisan, S., and Supavetch, S. “The Development of Web Processing Service Using the Power of Spatial Database.” *International Conference on Emerging Trends in Engineering and Technology (ICETET)*, Nagpur, India, 2009.

[4] Echtibi, A., Zemerly, M.J., and Berri, J. “A Service-Based Mobile Tourist Advisor.” *International Journal of Computer Information Systems and Industrial Management Application (IJCISIM)*, (2009): 177-187.

[5] Goodchild, M. F. “The Use Cases of Digital Earth.” *International Journal of Digital Earth*, 1(1), (2008): 31-42.

[6] Guanhua, C., Kunqing, X., Xiujun, M., Yanfeng, S., Yuanzhi, Z., and Lebin, S. "G-WSDL: A Data-Oriented Approach to Model GIS Web Services." *IEEE International Geoscience and Remote Sensing Symposium*, Seoul, Korea (South), 2005.

[7] Nadine, A. "Chaining Geographic Information Web Services." *IEEE Internet Computing*, 7(5), (2003): 22-29.

[8] Open Geospatial Consortium Inc., “OpenGIS Web Feature Service (WFS) Implementation Specification.” Version 1.1.0, Document number OGC 04-094, 2005(a).

[9] Open Geospatial Consortium Inc., “OpenGIS Filter Encoding Implementation Specification.” Version 1.1.0, Document number OGC 04-095, 2005(b).

[10] Open Geospatial Consortium Inc., “OpenGIS Web Map Service Implementation Specification.” Version 1.3.0, Document number OGC 06-042, 2006.

[11] Open Geospatial Consortium Inc., “OpenGIS Web Processing Service,” Version 1.0.0, Document number OGC 05-007r7, 2007.

[12] Peltz, C. “Web Services Orchestration and Choreography.” In *Computer*, vol. 36, no. 10, 46-52, IEEE Computer Society, 2003.

Author Biographies

Soravis Supavetch received his B.Eng degree in Survey Engineering from Chulachomklao Royal Military Academy and M.S. (Forestry) degree from Kasetsart University, Thailand. He is currently a PhD candidate in Survey Engineering at Chulalongkorn University, Thailand and currently attached to Royal Thai Survey Department. His area of interest includes service oriented architecture and geospatial processing service in geospatial information system.

Sanphet Chunithipaisan received his B.Eng and M.Eng degree in Survey Engineering from Chulalongkorn University, Thailand. He holds a PhD in Geomatics from University of Newcastle upon Tyne, UK. He is currently working as Assistant Professor in Department of Survey Engineering at Chulalongkorn University. His area of interest includes spatial database and data interoperability in geospatial information system.