The Tellus project. Remote GIS data editing and sharing using Open Mobile IS and gvSIG Mobile*

Manuel Gomez[1] Philippe Delrieu[1] Juan Lucas Domínguez[2]
mgomez@ubikis.con pdelrieu@openmobileis.org jldominguez@prodevelop.es

[1]Ubikis/MD69 Solutions – CEI 3, 62 boulevard Niels Bohr
BP 2132 69603 Villeurbanne, France – http://www.ubikis.com

[2]Prodevelop SL – Plaza Don Juan de Villarrasa 14, 5
CP 46001 Valencia, Spain – http://www.prodevelop.es

November 2009

Abstract

This paper shows that Open Mobile IS is a valid option to effectively edit and share remote GIS data in collaboration with gvSIG Mobile on a variety of mobile platforms. Open Mobile IS provides a series of very interesting features (possibility to work in disconnected mode, synchronization engine, encryption. . .). Virtually any type of database or storage system can be synchronized. In one of the tests, for example, we have used the *de facto* standard for free open source GIS databases (PostgreSQL/PostGIS).

Keywords: Open Mobile IS, gvSIG Mobile, synchronization, GPS, Java

1 Introduction

The Tellus project was born from a will to position in the mobile GIS market, in a complementary way to the classic connected systems. The Tellus project is based on the framework Open Mobile IS and the open source project gvSIG Mobile. It appears as an alternative for professionals who need a guarantee of data access with or without network.

*Visit its website: http://tellusproject.blogspot.com

Indeed, the idea is to supply a set of layers and components allowing to solve problems of mobile applications deployment in connected/disconnected mode binding cartographic layers. Example: mobile maintenance application (list of interventions, report edition, machine park. . .) with a cartographic support (custumer/intervention visualization, problems, intervention request, unplanned incidents. . .)

Tellus has three main ways of development:

- A real mechanism of map synchronization based on the Open Mobile IS synchronization engine to resolve the different types of conflicts which could occur. It also provides optimized data transfers based on the BitTorrent protocol.
- Providing a framework of development to allow interactions and communications between gvSIG Mobile and an Open Mobile IS application.
- Offering a real security for GIS data during the synchronization process and on the mobile tool.

If Open Mobile IS allows companies to deploy mobile applications as an extension to their Information System (IS), Tellus suggests completing this management by offering the possibility of widening also its Geographical Information System (GIS) in mobility situation.

For that purpose, Tellus bases itself on gvSIG Mobile.

2 Two independent projects for mobile devices: gvSIG Mobile & Open Mobile IS

The two projects involved in this paper are briefly described in this section.

2.1 gvSIG Mobile

gvSIG Mobile[1] is a JME CDC mobile GIS application that offers a reduced version of the functionalities of gvSIG[2]. The results presented here are part of the R&D efforts made by the author of this paper on a variety of mobile devices[3].

2.2 Open Mobile IS

Open Mobile IS[4] is an open-source project (GNU LGPL v. 2.1) that aims at providing all the necessary tools, API, and documents which enable effective mobile applications

[1]Official website: http://www.gvsig.gva.es/eng/gvsig-mobile

[2]Official website: http://www.gvsig.gva.es

[3]Unofficial gvSIG Mobile for Linux-based handhelds: http://gvsigmobileonopenmoko.wordpress.com

[4]Complete details of the Open Mobile IS framework in its website: http://www.openmobileis.org

development. It was written in Java by Philippe Delrieu (Ubikis[5], Lyon, France) and is mainly intended for JME CDC applications (pocket PCs and smartphones), though it's currently (Autumn 2009) been successfully adapted for CLDC devices (Java-enabled mobile phones). It is based on the Open Mobile Alliance standards (also known as SyncML) for synchronization purposes and is listed as a mature project in the OW2 Consortium[6].

2.2.1 Outstanding components in Open Mobile IS

Among other components, the Open Mobile IS framework provides:

Embedded database Open Mobile IS embedded database is a simple object database (FODB, Fast Object Database) optimized for low CPU and memory consumption. It provides a simple API based on SODA[7] to store and retrieve objects from the database. The object storage capability makes it more Java-friendly because it avoids mapping issues. The FODB consists of a series of collections stored as files. Each collection is a set of objects of the same type. Several types of indices can be used to improve performance.

Synchronization engine Synchronization is done between a table (or closely related tables) in the remote database (in this case, a PostGIS DB containing geographic data) and an object (or closely related objects) in the FODB located in the mobile device. Each field in a row of the table (server side) corresponds to a field in an object instance (client mobile side). Changes from the previous synchronization are applied on the other side. Both sides need to know which features have been modified since a certain moment in the past. To achieve this, incremental *sync. numbers* are used instead of dates, so that both sides need not have consistent system clocks. After each synchronization, the affected objects and rows on both sides are associated with a new sync. number, which will be compared with the sync. number provided by other clients in future synchronization requests. Objects or rows with a bigger sync. number have been modified more recently and must be updated on the other side. If there is a conflict (both sides have modified the same client-side object or server-side row), a pre-defined *conflict resolver* will decide which side prevails.

[5]http://www.ubikis.com

[6]ObjectWeb2 is a global open-source software community which goal is the development of open-source distributed middleware, in the form of flexible and adaptable components: http://www.ow2.org

[7]Simple Object Database Access: http://sourceforge.net/projects/sodaquery

3 Tellus or how to integrate Open Mobile IS in the GIS sector

3.1 Sharing/synchronizing GIS data

This section describes how GIS data can be synchronized using the Open Mobile IS capabilities. This architecture was created and tested[8] by Manuel Gomez (Ubikis, France).

It offers a service of dialogue between both systems (Open Mobile IS and gvSIG Mobile) based on the SOAP protocol, allowing a multitude of interactions between GIS projects (automatic project load, distant edition, information recovery in the IS, remote location), as well as an engine “of dressing” for gvSIG Mobile according the needs (IHM customisation via an XML file). Also a synchronization engine of GIS maps coupled with the synchronization engine of IS datas. The synchronization can have two modes:

- Via HTTP, cartographic files are exchanged in an express and global way (the cartographic project will be downloaded as an archive in one step).
- Via Bittorrent, cartographic files are given on a Tracker. Then the maps recovery is realized by the PDA on the tracker. Bittorrent allows an indispensable management of the connections/disconnections during a big data volume transfert (cartographic files) and a multiple sharing of the information on the whole park of PDAs (a PDA is in the same time customer and data server).

3.2 Several clients with write access: locking versus synchronizing

When several clients have write access to the same table in a database, it’s necessary to deal with concurrency issues in one way or the other.

Synchronizing is especially convenient when we expect a small number of conflicts among clients (optimistic approach). As we are doing row-oriented synchronization, a conflict means concurrent access to the same row in the same table. The well-known disadvantage of locking consists in the cost of waiting for locks held by other clients.

The Open Mobile IS approach is as follows:

- Optimistic approach (no locks). Database updates are performed as they come, asking help from the conflict resolver when needed.
- Each synchronization request consists in a series of *atomic* updates of single rows.
- More than one synchronization request can be processed simultaneously, so the first atomic update of request A can be followed by the first atomic update of request B, then the second atomic update of request A, etc.

[8]See details and available demo: http://tellusproject.blogspot.com

- This means that at the end of a synchronization request, both sides (client and server) don't necessarily hold the same information, but both sides are consistent. That unlikely divergence will eventually disappear in another synchronization process.

3.3 Feature-oriented edition via synchronization & disconnected mode

After defining an editable layer in the application based on the data available in the FODB, we can use the synchronization to effectively edit a remote PostGIS database (see an overall diagram in figure 1).

The FODB consists of a series of files stored in the mobile device file system, so the user can work in disconnected mode or even turn off the device and perform the synchronization whenever the internet connection is available.

Figure 2 illustrates how it works. Actions inside a red rectangle are started by the user via the GUI. In point A, the layer data visible in the map and the content of the FODB are equal. In point B application data and the remote database are the same, except for the unlikely and temporary divergence commented before.

Of course, we'll need to create a FODB driver to integrate it as a new data source in gvSIG. Figure 5 shows how this is done.

As you can see in the left side of figure 5, an array of cached updates is kept by the FODB wrapper, so each drawing of the layer does not need to query the files that make up the FODB. The *flush* option consolidates the edits performed by the user into the FODB.

We can also see the data as available in three different layers, as seen in figure 3.

3.3.1 Server-side architecture

Figure 4 shows the interface used on both sides to map the PostGIS table as a Java object. The geometry field is stored as a `String` in the object model. On the server side, the mapping of these fields is done with simple SQL sentences as shown in figure 7. The corresponding PostGIS table can be seen in figure 6. A small Java application runs on the server side, listens to client requests, computes the needed updates on both sides, performs the changes in the database and sends a response to the client telling which features need to be updated.

3.3.2 Proposed use

This feature-oriented syncronization mightbe useful in a variety of sectors:

- Urban facilities management, road incident reports, etc.
- Rural area monitoring, forest guards, census of valuable species.
- Accurate transportation tracking.

4 Conclusions and future work

The synchronization capabilities of Open Mobile IS can be easily integrated in gvSIG Mobile and allow the user to immediately perform remote GIS data edition. If the remote geodatabase is used by some kind of map server, then those changes are instantly visible world-wide.

Of course, those data can be managed and modified by other third-party applications, but in that case they'll have to take into account the management of the sync. numbers commented before, so that the synchronization feature does not find inconsistencies.

This first test has been done with a simple point layer. Since we have used the WKT format for geometries, it would be easy to do the same with line or polygon features.

Open Mobile IS also provides user authentication and management and encryption, as it's obviously intended for professional use.

Final results, software and videos can be found on-line[9]. Possible future work includes:

- Creating bundles for the upcoming version of gvSIG Mobile:
  - a bundle to add a FODB driver and synchronization client.
  - a bundle to allow interaction with gvSIG Mobile via SOAP by adding a small webserver (also provided by the Open Mobile IS framework).
- Synchronizing objects and remote tables if you know their structure only at run time.

[9]http://gvsigmobileonopenmoko.wordpress.com
http://tellusproject.blogspot.com

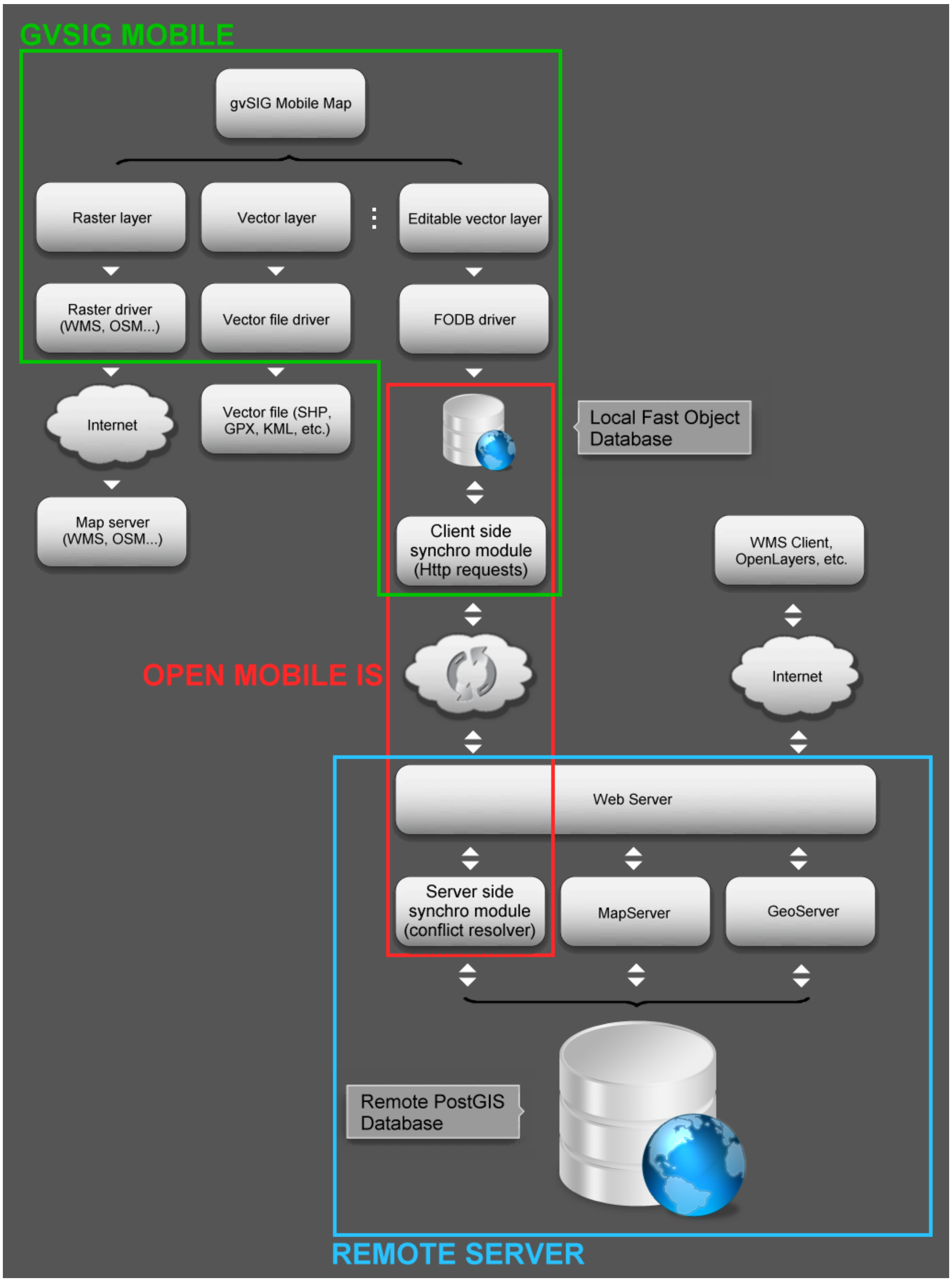


Figure 1: Overall diagram showing Open Mobile IS role.

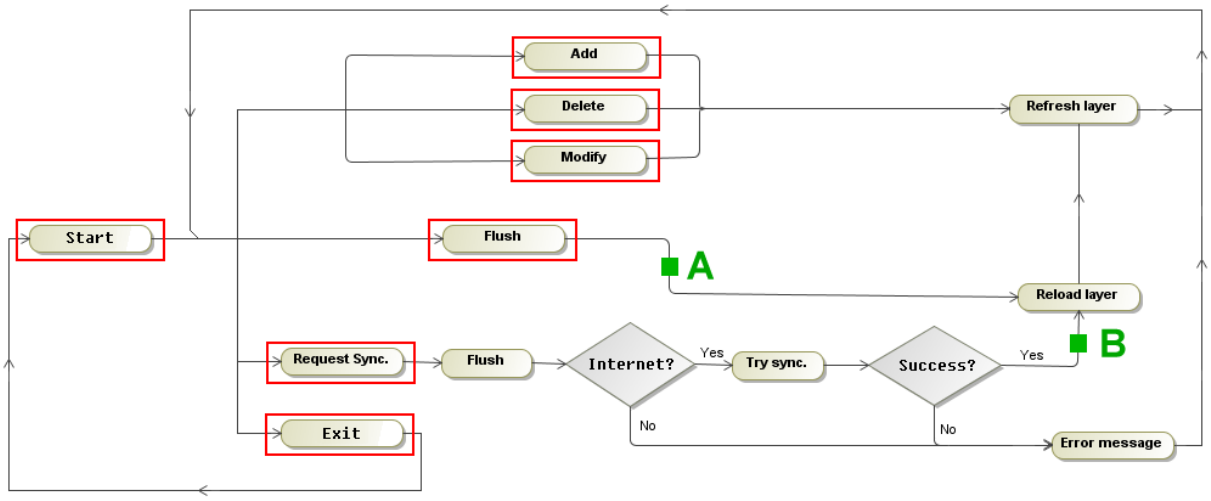


Figure 2: Work-flow using the synchronization capabilities.

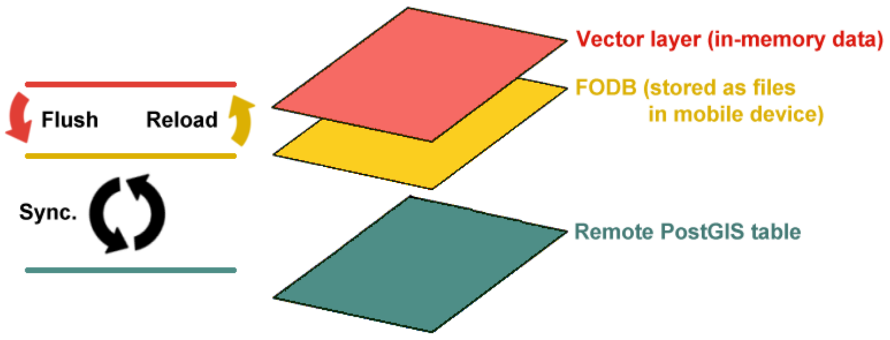

Vector layer (in-memory data)
FODB (stored as files in mobile device)
Remote PostGIS table
Flush
Reload
Sync.

Figure 3: Data flow seen as a three-layer interaction.

```
public interface Intervention {

    public String getIDInter();
    public String getAmenity();
    public String getName();
    public String getAge();
    public String getSize();
    public String getGeometry();

    public void setAmenity(String str);
    public void setName(String str);
    public void setAge(String str);
    public void setSize(String str);
    public void setGeometry(String wkt);
}
```

Figure 4: Java interface used in the object model.

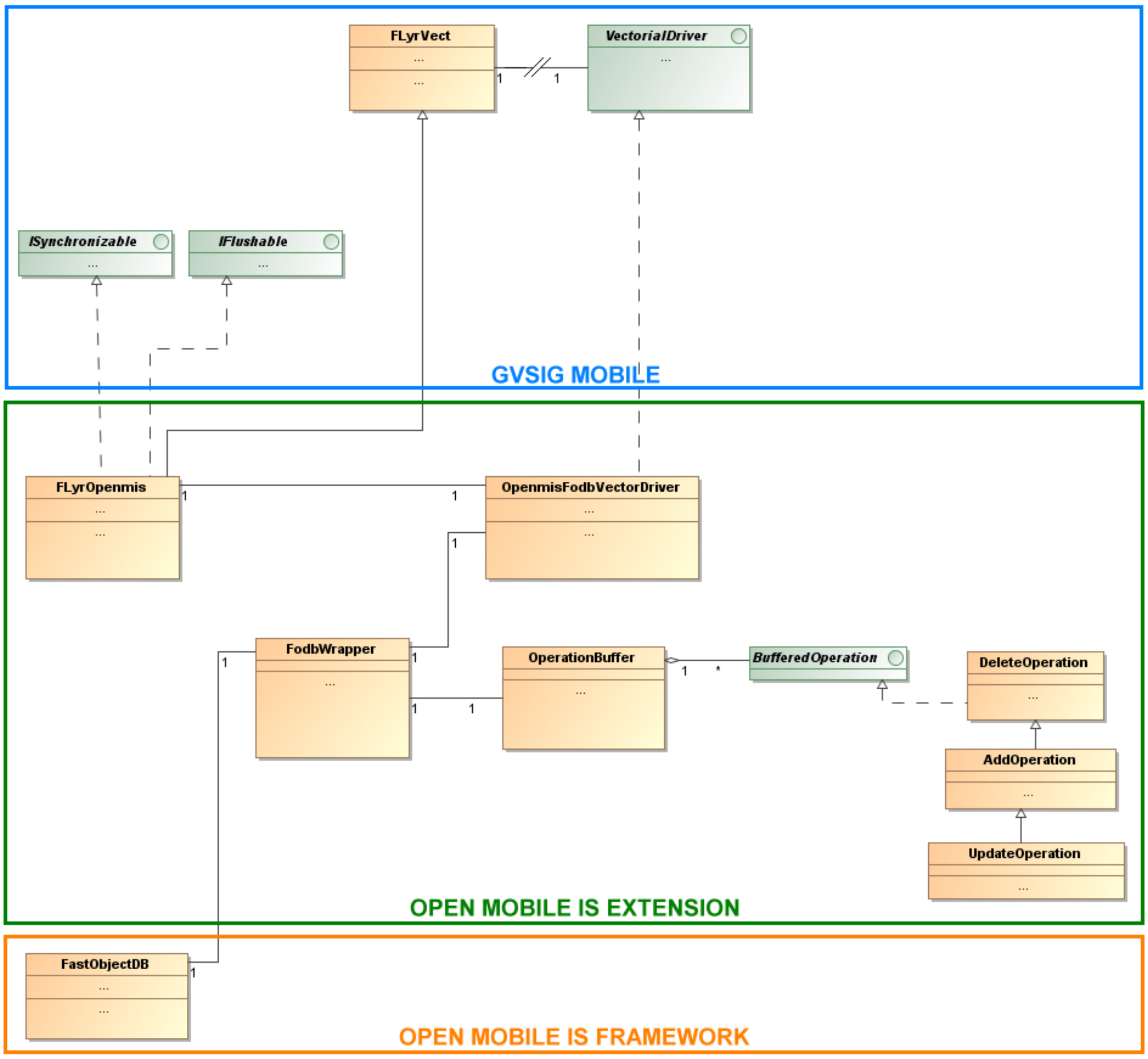
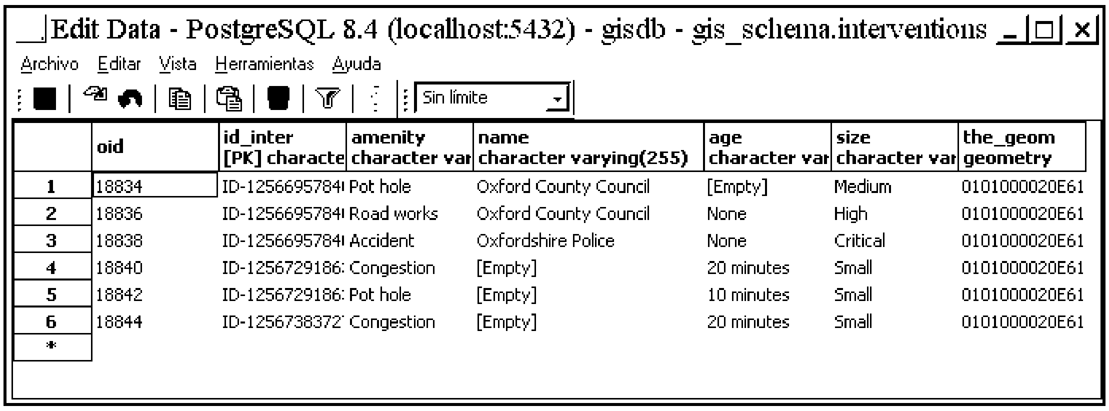


Figure 5: Simplified class dependency diagram.


Edit Data - PostgreSQL 8.4 (localhost:5432) - gisdb - gis_schema.interventions
Archivo Editar Vista Herramientas Ayuda
Sin límite

	oid	id_inter [PK] characte	amenity character var	name character varying(255)	age character var	size character var	the_geom geometry
1	18834	ID-1256695784	Pot hole	Oxford County Council	[Empty]	Medium	0101000020E61
2	18836	ID-1256695784	Road works	Oxford County Council	None	High	0101000020E61
3	18838	ID-1256695784	Accident	Oxfordshire Police	None	Critical	0101000020E61
4	18840	ID-1256729186:	Congestion	[Empty]	20 minutes	Small	0101000020E61
5	18842	ID-1256729186:	Pot hole	[Empty]	10 minutes	Small	0101000020E61
6	18844	ID-1256738372'	Congestion	[Empty]	20 minutes	Small	0101000020E61
*							

Figure 6: PostGIS table containing some data. Each attribute corresponds to a field in the object model

```
public class PostgisInterventionSyncServiceQuery extends AbstractQueryManager {

    protected static String createTableInter_ = "CREATE TABLE gis_schema.interventions ("
        + "id_inter varchar(20), "
        + "amenity varchar(255), "
        + "name varchar(255), "
        + "age varchar(255), "
        + "size varchar(255), "
        + "CONSTRAINT pk_id_inter PRIMARY KEY(id_inter)" + ");";

    protected static String queryGetInterventions =
        "SELECT id_inter, amenity, age, size, name, AsText(the_geom) FROM gis_schema.interventions";

    protected static String queryCreateInter =
        "INSERT INTO gis_schema.interventions (id_inter, amenity, age, size, name, the_geom) " +
        "VALUES ('%0%', '%1%', '%2%', '%3%', '%4%', %5%)";
```

Figure 7: These SQL sentences map the fields from the object model into the table's fields.