SIT-IN on Heterogeneous Data with Java, HTTP and Relations

M. Paolucci, G. Sindoni, S. De Francisci, L.Tininini
ISTAT
{paolucci,sindoni}@istat.it

Abstract
This paper describes the technological architecture under implementation in a project called SIT-IN. Aim of the project is the integration of a number of legacy territorial systems in a new spatio-temporal system, for providing a territorial reference framework to statistics users and producers. The architecture is based on a flexible and scalable central component implemented in Java and supported by a self-describing relational database. A common relational repository enforces coherence between the territorial objects and their spatial representations. A prototype has been fully implemented and is currently being extended to become the first release of the final system.

1. SIT-IN Overview

ISTAT, the Italian National Statistical Institute, is currently challenging the issue of integrating its various and different legacy spatiotemporal data collections. The ongoing SIT-IN (acronym for Integrated Territorial Information System, in italian) project started with a detailed feasibility study and has just released a first prototype, which has been built to ensure the feasibility and soundness of the chosen methodologies and approaches. The prototype development relied on Java and relational technologies in order to cope with data heterogeneity. We expect the results of the project will allow:

- easier fruition of spatiotemporal data for the corporate users;
- integrated usage of user-owned and system-owned data, with the territory acting as the interconnecting texture;
- coherence of spatial classifications in their time evolution;
- a set of guidelines, both methodological and technological for the development of the future spatio-temporal information systems of the Institute.

This paper will show the current results of the project, with special attention to technological choices and solutions.

The systems chosen for the initial integration experiments include SISTAT, the territorial history database system, providing information about the evolution in time of territorial administrative entities; CENSUS, the Institute GIS, providing the cartography of the Italian territory, down to the census tract level; BDT, a territorial datawarehouse of statistical data, originally built for data diffusion purposes; and finally SISTER, an address normalizing-geomatching system.

One of the main goals of the SIT-IN project is to define guidelines for database integration activities involving coexistence with and migration of legacy applications. Our approach to the problem, from the physical data integration point of view, is based on a loosely coupled federation of databases [4]. The federated systems continue to exist individually, and are connected through a dynamic data linking mechanism, which somehow resembles the lightweight alliances approach presented by the authors of [2].

The database federation is realised according the following principles.

1. Legacy systems are independent from the federation and are fully and totally responsible for the realisation and maintenance of their export scheme, that is, for what they share with the other federated systems. For each overlapping concept, only one of the component systems was chosen as the "owner". For example, the system responsible for the data about the administrative history of the Italian territory exports a *snapshot* to the federation and is responsible for its update. The snapshot is used by the federation as the only repository of data about the territorial hierarchy and its time evolution. This concepts separation has been possible only because the component system were well defined in number and type, and that their areas of competence were quite well defined, while only small, clearly identifiable parts were semantically interrelated or overlapping[1].
2. Time and space are the main access dimensions for any federated database. For example the territorial datawarehouse system is accessed through a set of *navigational metadata*, which allow the user to dynamically define the structure of statistical tables. At run time, the visualised report table refers to the chosen year and set of territorial entities (for example a set of towns).
3. Data coming from different physical worlds are integrated using relational technologies. For example, the geographical legacy database is run by ESRI Arc/Info, but has been migrated into SDE (spatial Data Engine), the ESRI spatial indexing system which allows to store and efficiently access geographical data through a relational database.

The resulting federated database is though heavily centred on the census geographical system. The database has been extended to accommodate enough semantic power to completely describe all the possible mutations in time of the objects of interest, that is, object birth and destruction, object modification (in borders, name, and ISTAT territorial codes), object inclusion in a less-grained object.

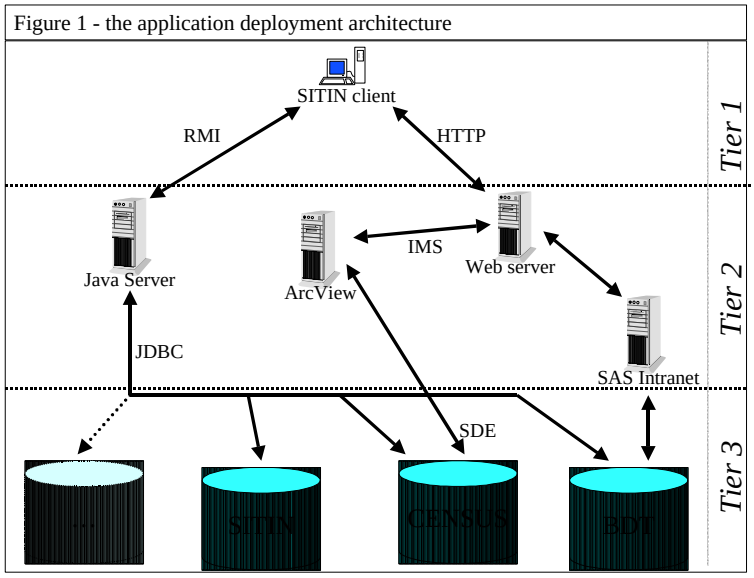
The key technological requirements of the project are:
- to experiment and propose scalable integration technologies;
- to enforce software reusability, by moving from objects to components;
We were also interested in developing a deep knowledge of the used technologies, so that this knowledge could be reused. This is a sound advantage both at the local level for the project developers, who work on a project which is likely to be subject to many changes and adaptation during its lifetime, and at the general enterprise level: the enterprise will acquire a substantial advance in technology control and standard issuing.

Following these principles, we chose Java as the development language for most of the in-house developed parts of the system. Other key advantages of the Java language were the availability of timely online support from SUN's Web site [6] and various public forums. Moreover, for the development of part of the user interface we were able to encapsulate and extend a publicly available third party Java Applet, ESRI's MapCafè, provided with ArcView IMS[1]. We chose Oracle as the RDBMS platform, mainly because three out of four of the databases to be federated were already stored in Oracle instances.

2. The SIT-IN architecture

The high complexity of the application context did not allow us to implement a simple client-server system. In fact, the SIT-IN prototype has the three-layer architecture depicted in Fig.1.

Figure 1 - the application deployment architecture



Tier 1

The data level is composed by a few Oracle instances. In particular, on top of the CENSUS database runs SDE (Spatial Data Engine), which allows for the efficient access and manipulation of spatial data.
Implementing the above declared principles required a massive usage of mediators [3] to wrap and interconnect the different areas of the systems. The use of Java and JDBC allowed us to easily implement the mediator components for each member of the federation. In particular, for each database in fig. 1, a Java mediator module has been developed, which fully relies on the functions provided by the Java server module. The necessary effort to develop the mediators was extremely reduced: in fact, the bigger one is implemented by only 300 lines of code even if it provides a sophisticated method to mine historical information about the territory transformations.
Another particularity of this approach is that, thanks to the independence of the layers, the number and kind of accessed Oracle instances is not predetermined at start-up time. This feature allows users to dynamically link their own Oracle data to the system, in order to perform administrative checking and spatial analysis on them. As an example of such a

[1] ArcView is a lightweight GIS and IMS (Internet Map Server) is an extension which allows for cartography serving on the Web; MapCafè is the applet client to the IMS.

mechanism, we implemented a function for dynamic geographic theme building and visualisation on users' data. By providing all the necessary connection information, a user is able to dynamically integrate an external database table containing a set of statistical values which refer to a set of territorial entities. The integration is made possible by the existence of a domain specific piece of information: the territorial code. This is a standard code which is used by our institute to identify the administrative territorial entities. This feature and the relational storage of spatial data, allows for the retrieval of the geographical representation of the entities by the spatial engine and the consequent visualisation of a thematic map of the territory.

The dynamic integration process is implemented by a *meta-querying* system [5] which is part of the SIT-IN metadata management system, a set of database tables containing data about users, linked external tables and values and, above all, data about the administrative hierarchy of the Italian territory. To each administrative level, a parametric query is associated which builds an integrated temporary view out of a user's table containing statistical data with the corresponding spatial granularity. The metadata-driven approach is very useful not only because it allowed us to easily implement the dynamic integration process, but also because we believe it is a very open architecture. In fact, one of the current development lines of the project is the extension to the support of different territorial hierarchies.

Tier 2

The application layer is the most complex layer and is composed by a few different systems. The Java server connects the client layer with the alphanumeric data sources, so decoupling the interface from the Oracle access; in this way, JDBC is only used[2] by the server, while the user interface receives data through RMI (Remote Method Invocation), as Java vectors of objects.
The other main communication path is used to retrieve spatial representations of objects: it starts with a connection between the user interface and the web-server, which invokes a CGI, which in turn connects to ArcView IMS and through it to SDE and eventually to the polygons. Communication between user interface and the cartographic server passes through an HTTP stadium and may seem to be, for this reason, stateless. The HTTP stadium was not avoidable so far, because we needed to use (and extend) MapCafè, a free visualisation applet provided together with IMS, which is configured in that way.
So we had to turn this communication architecture from stateless to stateful. To this purpose we used a (popular?) trick yet employed by MapCafè. It consists of masking data relative to the state as URL parameters in a HTTP-GET connection, and as hidden commands in the answer. We extended this mechanism to fit our needs.

Last but not least, we use SAS Intranet for the presentation of statistical data on the Web. We chose to introduce it in our architecture for its HTML table formatting features but also to experiment the ease with which an external, HTML-driven system could be integrated in the framework.

Tier 3

The client user-interface is a Java applet, developed with version 1.1.8 and swing components. To use the applet on a generic browser, because of the presence of swing components, the user is required to install a compliant plug-in.

A characteristic that strikes the eye of our architecture is that there are two distinct communication paths: the "geographical path", centred around ArcView IMS, and the "relational path", which is accessed from the client through the Java server. The only connection point between the two paths is at the client level; even if the spatial data reside on the same database as the alphanumerical ones.
The coherence of data, independently from the design of connections, is assured by the integration at the data base level. Thanks to this fact, we have been able to safely choose to maintain the two utterly distinct communication pipes: the first (JDBC-RMI) offers to the user the possibility to select data of interest, maintaining hidden references to the geometrical data; the second, which is activated through an explicit user request, connects to the geometric engine to perform specific geometric operations and to retrieve the graphic results. Since this second set of operation is substantially more computationally complex than the first one, complete separation gives us a relevant advantage: it is then possible to accurately separate processing times and also to substitute the entire geographical processing system whiteout touching the selection systems and a major part of the interface.

3. Learned Lessons

Throughout the above design and development experiences we learned many different lessons. Here we briefly comment on what we think are the most interesting ones.

[2] And imported, allowing the client to be thinner.

Having a Java server is useful.
The use of Java to develop server side intelligence allowed us to easy implement a clean role decoupling mechanism between database access and data elaboration on the server side, and presentation of fully object-encapsulated data on the client side. We found also that, while Java contains specialised constructs for data access like recordsets and the similar, there is little standardisation in the persistent object exchange area, especially when using version 1.1[3].
The more you put on the database, the less code you write.
One of the most radical paradigm we followed, is to push as much semantics as possible in the database. The use of metadata and meta-queries is an example of such a philosophy. Its main advantages are: exploitation of the reusability of code (same code implements different semantics through the use of parametric queries); easier extensibility of the system (adding new meta-information is easier than writing new code); optimal distribution of responsibilities (changes in the database schema propagate only to changes to the meta-information, which can be done by a data administrator).

References

1. D. Fang , J. Hammer, D. McLeod, The Identification and Resolution of Semantic Heterogeneity in Multidatabase Systems, Proc. of International Workshop on Interoperability in Multidatabase Systems, Kyoto, April 1991
2. Roger King, Michael Novak, Christian Och, Fernando Vélez: Sybil: Supporting Heterogeneous Database Interoperability with Lightweight Alliance. NGITS 1997: 0-
3. Gio Wiederhold: Mediators in the Architecture of Future Information Systems. IEEE Computer 25(3): 38-49 (1992)
4. Amit P. Sheth, James A. Larson: Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. ACM Computing Surveys 22(3): 183-236 (1990)
5. Frank Neven, Jan Van den Bussche, Dirk Van Gucht, Gottfried Vossen: Typed Query Languages for Databases Containing Queries. Information Systems 24(7): 569-595 (1999)
6. http://java.sun.com

[3] We decided to use version 1.1 anyway because of the wider availabilty of VMs, both on OSs and on browsers.