Small scale integration problem: pop census

Antonio Paolacci

July 8, 2011

1 The problem

The national institute of statistics (NIS) of a fictitious country is organizing its the census of population. The main innovation they want to introduce is the possibility of compiling the questionnaire through a online web application, besides the ordinary paper questionnaire, which is collected on the field by a network of territorial operators. The IT department in the NIS has to develop the whole information system supporting the census operations. The system has two main components:

- Management application: web application used by operators to organize the on-field collection work of paper questionnaires. It shows the status of completion of eacn questionnaire.
- Online questionnaire: web application accessible by each of the 40 millions family in the country.

Being developed by two different teams the two applications rely on different databases. However, they must be integrated in two ways: 1) the online questionnaire must access the list of families for checking the login information; 2) the online questionnaire application must update the status of completion when a questionnaire is completed. They will be deployed in the same data center. The paper questionnaires can be returned after compilation to the local postal office. Questionnaires are unreturned after a certain period of time are collected door-to-door by operators. Then, there are two ways for updating the return status of a questionnaire in the database:

- Integration with the information system of the postal company. This system can provide the return status of all the questionnaires. Design the integration solution considering the tradeoff between data size and frequency of updates.
- Operators carrying out the door-to-door collection are provided with mobile phones with an application through which they can update in real time the collection status while they are on-field.

Design the architecture of the whole system and detail the various protocols for updating the questionnaire status.

2 Requirements review

It's known by the requirements, that management application and online questionnaire application are deployed on *different database*, so I assume that in the database of management application (then simply called **db1**) are stored families login informations, return/updated

status of all questionnaires, territorial operators login informations, postal office login information... Also I assume that the online questionnaire application database (then simply **db2**) only collects data about filled questionnaires and nothing about status of completion; for this reason the online questionnaire application does commit operation to db2 only if db1 does commit updating the return-status of a questionnaire. For the user this is a synchronous behaviour, so the user waits to complete succesfully his questionnaire only if both commit are done. This behaviour assumption can be reasonable because we are in the *same data center* and there are no big network bandwidth problems and low latency. An alternative (not implemented) could be represented by an asynchronous method that updates status informations from db2 to db1 periodically, in this case db2 stores status informations. Db2 and db1 are inconsistent for a while.
The requirements show that update-status operations of a questionnaire come from different source:

1)online questionnaire application when a new filled questionnaire complete successfully

2)information system of the postal company if there is an integration system with NIS system

3)mobile phone of operator who collects door-to-door unreturned questionnaires

Due to this specification I assume to split the *update-status end point* providing distinct web services. One end-point used by online questionnaire application like an *internal* web service, others end-points used by information system of postal company or mobile phone app of the territorial operators. The web services are divided according to different levels of access security (one come from internal LAN, others no and they require authorization mechanisms) and to different exchanged data size and frequency of updates.

3 The logical architectural design

The Figure 1 shows all the significant components in the architecture. Management and online questionnaire applications are *web application* which rely on application server for example, accessible from outside the NIS-data-center respectively by operators and families. The PC-monitors at lateral sides indicate a client, simply *web browser*. Databases are separated because application have separated purposes and this represents a small internal integration problem. So the architecture needs a middleware layer and to deploy *four web-services end-points*: update-status for online questionnaire application, two update-status respectively for mobile phone and information system of postal company according to different data size and frequency update, one check-login to authenticate families to online questionnaire application or NIS operators,postal office,door-to-door operators to the management application. At the bottom, based on the adopted solution, it could be a mobile phones with an *application* through which door-to-door operators can update in real time the collection status of a questionnaire or an integration service with the IT system of postal company represented by a *client-server application* installed on every postal office of the country.

4 The technical architectural design

The Figure 1 shows the technical components and links of the infrastructure that IT departement at NIS has to develop. Both management and online questionnaire application are web application (*JAVA* and *PHP* respectively), so it's possible to iteract with them via *http-IP* protocols. Therefore the two PC-monitor in the diagram are a typical *web browser*, one represent 40 million of family that are selected or want to fill a questionnaire online, the other one represent NIS operators that are organizing and overseeing the collection of all,

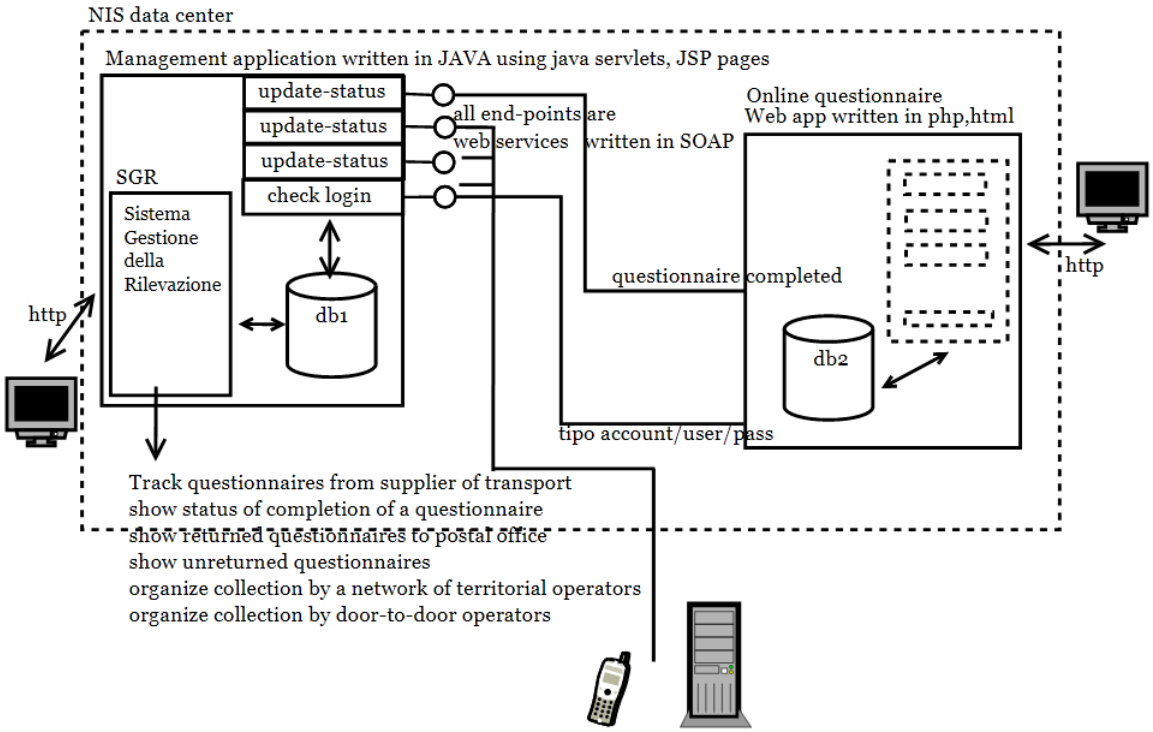

Figure 1: Diagram of all the significant components in the architecture

web and ordinary paper questionnaires. NIS operators can access to the management application from the intranet of NIS but also from outside so needs the same levels of security and authentication of online questionnaire application. The two applications rely on different databases. The database could be implemented with different technologies, *Oracle* or *MySql* for example, because they serve separate applications and necessarily we need an integration middleware to communicate. The *middleware* choosen is *SOAP*. In a SOAP-service requests are described by a language called WSDL so this means that in order to make best use of a SOAP-service, you have to be able to read an XML file that describes all the information needed to invoke operations. Having a rigid description of the service is a feature that in some scenarios can be an advantage, such as communication exchange in a large data center where information, for security reasons must be typed. A SOAP service routes requests to a unique address called *end-point*. The SGR (*Sistema Gestione Rilevazione*) software provide standard functionalities to show data stored in the *db1*. It's possible:
to track questionnaires from supplier of transport (postal company for example),
to manage sending questionnaires to the population,
to monitor returned questionnaires to the postal office,
to monitor unreturned questionnaires and organize collection by door-to-door operators,
to organize collection by a network of territorial operators.
It's also possible to update the state of a questionnaire with different solution end-points after a *check-login(account-type,user,pass)* function return positively. Online questionnaire application permits to login as a registered family (check login end-point) and to fill a online questionnaire based on *html forms* and *http-POST* method. Data about the specific questionnaire related to the specific family will be stored on the *db2*. A user completes successfully a web questionnaire only if the update-status end-point invocation returns correctly from db1 and data are correctly stored in db2. To do so we need a complex management

of transactions, expecially we need to implement and use the **Two Phase Commit** protocol. The normal transaction management is not sufficient when performing distributed transactions. The applications makes a change to the database db1, then make a change to the database db2, then want to confirm both. At this point a simple message commit does not allow proper management of the distributed transaction. In fact it's possible that one of the two databases, for whatever reason, has been an error and the data hasn't changed, the commit operation performed on the database would lead to an inconsistent state. Following in Figure 2 is shown the sequence diagram of the whole protocol submitting a web questionnaire. The 2PC protocol commits the data in two steps. In the first step a "coordi-

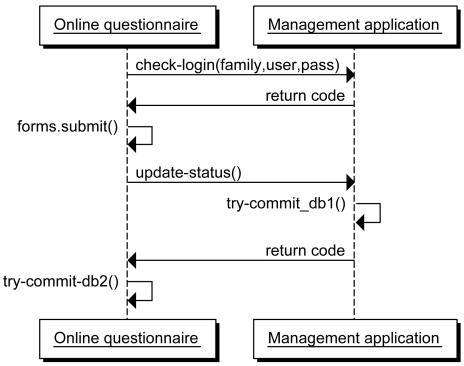

Figure 2: UML sequence diagram of a successfully complete questionnaire

nator" of the transaction sends the *PREPARE TO COMMIT* message to all the databases or agents involved in the transaction. If they all respond positively within the *timeout*, the coordinator sends the commit message, if someone responds negatively or does not respond, the *ROLLBACK* message is sent. The adoption of a 2PC protocol increases the number of

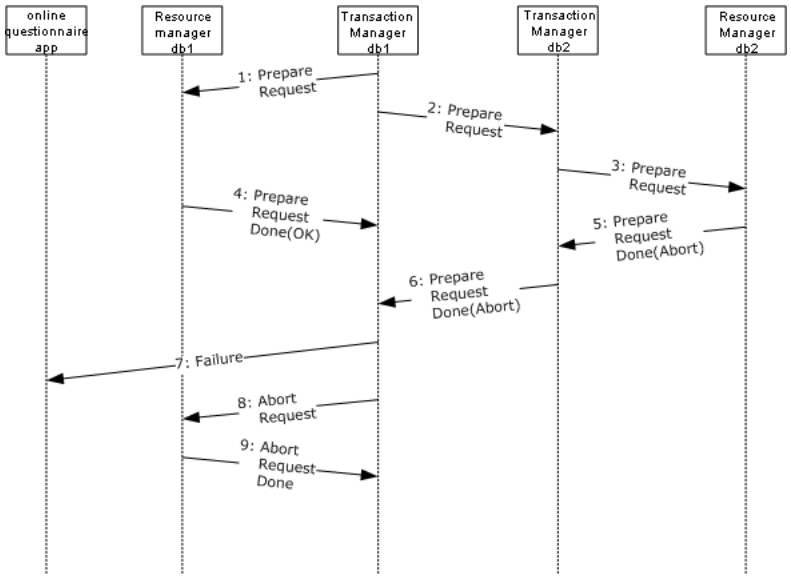

Figure 3: Sequence diagram of a aborted transaction

exchanged messages and response-time to the web user but, both DB are in the same data

center, so we expect not to have network bandwidth problems.
The bottom part of Figure 1 presents the solution at the problem "How data can be obtained from the supplier of transport/collection of questionnaires?", the choise is between:

1)a desktop application in a postal office or

2)a mobile phone application provided to a door-to-door NIS operator.

In the first case we have to integrate the information system of postal company with the management application. To do so management application implements a web service *update-status* that can be invocated by postal office client after authentication via *check-login(postal-office,user,pass)*, at closure time, to transfer data about the list of returned questionnaires and to prepare sending back answered questionnaires. This end-point differs to the others in terms of *amount of data*, *low update frequency*, *transfer method*. In this case we have in a *asynchronous* mode, a big *data chunk* to exchange and update into db1.
The second solution could be a mobile application but requires to manage a large door-to-door collection with many operators. The mobile app authenticates to management application with *check-login(mobile-operator,user,pass)* and connects to a different *update-status* end-point to update in *synchronous* mode the status of a questionnaire. In this case every changes in the supplier of transport state have to be notified to the NIS in real time.
Table below summarixes and shows significant trade-off among two solutions proposed.

	Transfer Mode	Data Size	Update Frequency	Update Time	Operators Number
1)Integration	asynchronous	big chunk	low	at closure time	small
2)Mobile app	synchronous	a questionnaire size	high	real time	big