

**COHERENCE MAINTAINANCE IN COOPERATIVE  
INFORMATION SYSTEMS:  
THE ACCESS KEY WAREHOUSE APPROACH**

Franco Arcieri<sup>1</sup>            Elettra Cappadozzi<sup>2</sup>            Paolo Naggar<sup>2</sup>  
                                 Enrico Nardelli<sup>2,3</sup>            Maurizio Talamo<sup>2,4</sup>

1. Consultant to AIPA for the SICC (“Sistema di Interscambio Catasto Comuni”) project.
2. “Coordinamento dei Progetti Intersectoriali” of AIPA - “Autorità per l’Informatica nella Pubblica Amministrazione”, Roma, Italia.
3. Dipartimento di Matematica Pura ed Applicata, Univ. of L’Aquila, L’Aquila, Italia. nardelli@univaq.it
4. Dipart. di Informatica e Sistemistica, Univ. of Roma “La Sapienza”, Roma, Italia. talamo@dis.uniroma1.it

M.Talamo is the CEO of the Initiative for the Development of an IT Infrastructure for Inter-Organization Cooperation (namely, “Coordinamento dei Progetti Intersectoriali”) of AIPA - “Autorità per l’Informatica nella Pubblica Amministrazione”.

**ABSTRACT**

In this paper we present and discuss a novel architectural approach supporting the integration among legacy information systems of autonomous organizations. It is based on the use of a data warehouse in a new conceptual role. Namely, we propose to use it, during the design and implementation phases of a cooperative information system, as a tool supporting the coherence maintenance of the underlying databases and the efficient management of accesses to them. Our approach is rooted in the SICC project for cadastral data exchange among Italian Municipalities, Ministry of Finance, Notaries, and Certified Land Surveyors. Research results reported here are an abstraction of solutions introduced in the SICC project and validated through the development of various inter-organization cooperative information systems, managed by the “Coordinamento dei Progetti Intersectoriali” of AIPA, the Italian Authority for Information Technology in Public Administration.

**Keywords:** cooperative information systems, interoperability, access management.

## **1. Introduction**

The development of cooperative information systems is an issue that has become more and more important in the last ten years [3], due both to the explosion of the

availability of network connections and to the continuously increasing presence in any organization of computerized information systems used in everyday activities.

In this field the really hard problems are almost never the technical ones but rather have an organizational nature. In fact, in the last years the availability of computing power and connectivity at affordable prices has reached levels that were incredible just twenty years ago. This makes it feasible to implement almost any technical solution may be needed. But the big challenge is to build a cooperative information system that is able to smoothly support the way organizations interact, being minimally intrusive and favouring a real and effective cooperation.

This is even more complex due to the fact that organizations have developed, during the last thirty years, their own information systems without thinking and designing them in terms of cooperation with entities outside their boundaries.

Now that real connectivity to the whole world is technically feasible and economically affordable, the interfacing among these legacy information systems, to support the way organizations interact, seems to be just a step away. In the reality, unfortunately, it proves much harder to obtain.

Clearly, in this context, existing information systems are physically distributed: this implies that the underlying databases are distributed and independently managed. But this is just a part of the whole picture, and not the most important one: the state of the art of research and practice in distributed DBMS is highly advanced [1].

The fact that existing information systems have been independently developed implies that used technologies have a wide range of variability and are, in many cases, rarely compatible and hardly able to interact. This is a crucial issue and the use of advanced technologies (e.g., Distributed Databases [1], Wrappers and Mediators [2]) can be of help in dealing with some of the technical problems, but it is deadly wrong to think that this is a first step to be anyway undertaken and that, once these technologies are put in place, the cooperative information system will be easily obtained by just implementing the needed interface layers and the overall layer of coordination.

One of the objectives that is hardest to reach in the development of cooperative information systems is the *coherence of the overall (distributed) set of data*. On one side, in fact, data are independently and autonomously managed by the various organizations. On the other one, data are needed and used also outside the organization producing/managing them and controlling their changes.

These clashing situations will produce incoherence in the overall set of data, sooner or later, with absolute certainty. Since the lack of coherence derives from an organizational problem, then the technical solution has to be designed so as to match needs and behaviour of the organizations involved. Moreover, the technical solution has to be designed in order to provide good performances of the overall system.

Hence the issue is by which means, during the design of a cooperative information system, a feasible solution to all the above requirements can be provided.

To address this problem, in this paper we propose and discuss a new architectural approach, namely the **Access Keys Warehouse** approach, to be used for the development of cooperative information systems integrating legacy information systems of autonomous organizations.

This approach proposes a novel role for the concept of Data Warehouse, namely suggests that a (new kind of) warehouse can be set up to guide and control accesses to the underlying databases. This allows to solve the coherency problem with good overall performances and to provide a methodological guidance for the development of a cooperative information system.

Our approach also establishes a high level framework for the definition and design of a technological solution to the problem of realizing a transaction manager for inter-organization cooperative applications interacting over Internet.

The approach is rooted in the SICC project [9, 11], started by AIPA in 1995 to deal with coherence maintenance issues in cadastral data exchange, as required by the law [8]. Further details about the SICC project are given in Appendix A.

Since then, the “Coordinamento dei Progetti Intersettoriali” of AIPA, the Italian Authority for Information Technology in Public Administration (PA), has successfully used solutions first introduced in the SICC project in the development of a number of inter-organization cooperative information systems, with the aim of defining an Information Technology infrastructure for inter-organization cooperation among PA organizations. This is within the more general framework of the development of an Information Technology and Telematics infrastructure (namely RUPA - “Rete Unitaria della Pubblica Amministrazione”) able to support, for the various PA organizations, both their internal work and service supplying to citizens [4].

Hence, beyond its theoretical value, the strength of our approach derives from having been validated through the implementation of real-life systems.

The structure of the paper is the following. In Section 2. we describe in detail the scenario our proposal makes reference to, while Section 3. discusses what has been proposed until now to design cooperative information systems in such a scenario. Next, the Access Keys Warehouse approach is presented at a general level in Section 4. and at a more formal one in Section 5.. Subsequently, Section 6. shows how to use it in some typical scenarios, while Section 7. gives a conceptual view of the system architecture and extensively describes its major components. Section 8. concludes the paper. In Appendix A we describe a real-life example, taken from the Italian PA, of the scenario introduced in Section 2..

## 2. The scenario

The scenario we consider is the following:

- there are many large and autonomous organizations which have to cooperate so that each one of them is able to reach its own goals, that are usually set by some external entity (e.g., a law, a statute);
- each organization has, since a long time, developed its own computerized information systems, which is maintained and expanded within the organization itself;
- an organization is not the source/manager of all the data it needs to reach its own goals: interaction with other organizations to obtain data that are

needed always happens through the exchange of certified messages; in some cases the interaction flow is specified by a law;

- there is no real possibility of enforcing and/or coordinating changes in the legacy information systems of the various organizations;
- there are scarce resources (i.e. time and/or money) available to be able to afford a re-engineering process, even for a single legacy information system;
- legacy information systems have to remain operational to continue providing their services to organizations.

The above described scenario is typical of the Italian PA (and possibly of the Public Administration of other countries). The large organizations are the various ministries (e.g., the Ministry of Finance) and public companies (e.g., the National Institute for Social Welfare). An example of a goal of an organization is to support an action of the Government (e.g., to help Government in deciding about changes in taxation rates for the various income levels) or to provide a specific service to citizens (e.g., to keep a citizen up-to-date with its rights and duties according to the relevant pension schema).

There are two aspects of the Italian PA making its situation a very hard case for the development of cooperative information systems:

- the various ministries and public companies have a long standing tradition of autonomy and independence, both at the central government level and at the regional level;
- there is a huge variety of organizational models, software and hardware architectures, and technological frameworks, with different change dynamics.

We are aware of no other organization in the world, having these characteristics, that has defined a strategy to solve in an acceptable way the problem of developing cooperative information systems in the described scenario.

Note also that the scenario might even be used to describe, in some cases, the situation of a multi-national company that has split up in a number of smaller companies for the purpose of being more agile and effective on the market.

In this scenario, to increase efficiency of the organizations and their effectiveness, it would be really useful to have cooperative information systems supporting their interactions. But such systems can be successful only if they realize some form of integration between the involved legacy information system.

To reach this goal, the most critical issue is how to enforce and maintain the coherence of the overall (distributed) set of data. The problem is that in this scenario there are certainly some data that are independently and autonomously managed by an organization, but that have to be necessarily used also outside it. Hence the problem is that an organization needs to use some data but has not a full control over them and the way they change.

A simple two-levels example describes this coherency issue.

Assume there are three organizations, namely *A*, *B*, and *C*, working on data sets that are partially shared at a conceptual level. Let us assume, for the sake of the discussion, that a single integrated conceptual schema of the whole set of data is available. It is clear that, since data are physically recorded in databases distributed among different organizations and independently managed, critical coherency issues arise when data change.

In particular consider, at the first level, the case where organization *A* communicates the change of value for data item *X* to organization *B*. Item *X* is usually recorded and managed by *A* while *B* just uses it. Organization *C* is interested in *X*, also, and, for its organizational purposes, has requested in the past its value to *A* and kept a record of it. It is clear that, even if at the current moment the interaction is between *A* and *B* only, *C* is certainly interested in such a change. But one can neither impose on *A* to communicate the change of *X* to every organization that requested its value in the past (take just into account the complexity of keeping track of all these queries) nor expect that *C* periodically polls *A* to ask if any update occurred to *X*. In this last case, in fact, beyond the additional burden that this correlation would impose on organizations involved, consider the highly critical issue of synchronization delay. This is potentially very dangerous, since it would make the value stored in *C* appear up-to-date with the original value in *A*, while it is not and the change has not yet been propagated.

But there is more. It may well be the case, at the second level of coherence maintenance, that attribute *X* has many conceptual relations with other data managed by *C* for its purposes. It may then happen that this update of *X* causes a mismatch between *X* itself and the other data to which it is bound within *C*. It is clearly infeasible that one expects *A* takes into account such a potential mismatch while carrying out its update but, on the other side, it is also clear that *C* has to have some means to efficiently correlate the change of *X* with the representation of the reality of interest it manages.

### 3. How the scenario is tackled in current approaches

The development of cooperative information systems in the above described scenario is currently based on a variable mixtures of the following approaches [5]:

- legacy information systems are wrapped using object oriented technology and for each of them some functions and data are made visible (or *exposed*) so that the legacy system can provide *services* to its outside,
- synchronization among legacy systems is provided by means of the *publish & subscribe* interaction protocol <sup>1</sup>,
- some *middle-ware* layer is implemented to realize coordination and cooperation functions by using as black boxes the exposed services provided by the encapsulated systems; in the most advanced proposals, functions in this layer are realized by referring to the concept of *broker* (or *mediator*), as an independent agent facilitating the interaction among the wrapped systems [5].

---

<sup>1</sup>It is not reasonable to use in this scenario other protocols, due to the quadratic number of interactions they develop in the worst case.

The main drawback of such an approach is that by considering legacy information systems as black boxes and then implementing coherence maintenance functions in the middle-ware layer using the exposed services the outcome may be catastrophic in terms of performances. In fact, for each change in attribute  $X$ , to which an organization is interested, the execution of coherence maintenance functions may activate many and many internal functions of the legacy information systems involved in the cooperative framework.

Given the wrapping approach, in fact, **access number and access paths required to a legacy information system by the execution of coherence maintenance functions are completely out of the control of the designer.** This is definitively not a good design practice and cannot be accepted.

The critical issue is that an interaction mechanism based on the publish & subscribe concept can be realized rather easily, even for legacy systems, by wrapping them using object oriented technology and implementing synchronization mechanism in the middle-ware layer. Unfortunately, this advantage of the wrapping approach is a serious drawback from an engineering point of view, in this specific case, since *it does not allow to evaluate and measure the impact on performances of the wrapped systems deriving from outside requests.* Hence it is not possible to perform a rightsizing of the overall cooperative information system with a cost-benefit approach and it is not, in general, possible to provide reliable performances independently of the costs involved.

Furthermore, an unrestricted use of the publish & subscribe mechanism in a scenario like the one described in Section 2., is going to severely affect the performances of the overall system. In fact, if any of the subscribers of a given class of data receives all updates to each elements of the class, this degrades the efficiency of the overall system whenever subscribers are more than some tens and/or an update is of a size of more than some hundreds of Kilobytes.

Before entering into the description of the **Access Keys Warehouse** approach, presented in Section 4., the reader may wish to have a look to a real-life example, in Appendix A, of the scenario introduced in Section 2..

#### 4. The Access Keys Warehouse approach: a general description

To effectively deal with the development of cooperative information systems in the above described scenario we introduce in this paper the **Access Keys Warehouse (AKW)** approach. This is a novel architectural approach allowing to smoothly develop, in a cost-effective way, cooperative information systems supporting interaction among autonomous organizations.

With this approach the designer of the cooperative information system has the possibility of evaluating, during the rightsizing phase, the impact of design solutions on performances and to define the system taking into account the outcome of such an evaluation.

We can informally describe, at a high abstraction level, the AKW approach by referring to the two-levels coherence maintenance problem introduced in Section 2..

Then, at the first level, a system implemented according to the AKW approach has a *coherence maintenance* role, controlling the evolution of the cooperative information system so as to ensure it is always up-to-date with the original sources of data.

The technical device making this possible is a mapping among the data items existing in the various distributed databases and that are of interest for the cooperation. This mapping is created and maintained by the system services of the AKW approach, and is realized through an *exchange identifiers database*. This is a *data repository* containing, **from a virtual point of view only**, all data items that can be found in various databases of a distributed system: from a physical point of view most data items remain at their locations.

The consequence is that, at the first level of the coherence maintenance problem, by using the exchange identifiers database one needs to propagate variations only for access keys and not for every data item changing its value in the cooperative information system.

The *exchange identifiers database* is physically built, but contains only *access keys* and *logical links* for data items in the various databases of the distributed system. The access keys are attribute names, selected from the existing attributes in the underlying databases: the main rule in order to select them is that their concatenation constitutes a unique identifier for the data item. Logical links provide the access paths to the physical (distributed) databases where further data elements about the identified data can be found.

Note that attributes in the exchange identifiers database act towards legacy systems as access keys: their value is used to query legacy systems. Hence they are **not** physical pointers, and the legacy systems maintain their independence and transparency both with respect to location and to implementation.

At the second level, a system implemented according to the AKW approach has an *access management* role, guiding accesses to the legacy systems referred by the logical links so that accesses are minimally intrusive, have a minimal impact on their performances, and correlations required for coherence maintenance among the legacy systems can be efficiently executed.

The technical device making this possible is once again the *exchange identifiers database*. In fact, in its design, attribute names have been selected according to two criteria:

- the set of selected attributes has to be small enough so that its materialization can be efficiently managed and queried;
- the set of selected attributes has to be large enough so as to be able to contain all access keys needed to deal with coherence maintenance issues among legacy systems while keeping their performances at an acceptable level,

With respect to data involved in the cooperative application, we can identify, broadly speaking, **Suppliers** and **Users** of data. A Supplier is any entity generating a data item and/or entitled to change it, while a User is any entity interested to use a data item for its own purposes. The exchange identifiers database keeps record of which are the Users and the Suppliers for the various data items.

The exchange identifiers database is populated using data existing in the various distributed locations. Values of access keys are supplied by the organizations involved in the interaction while the correlation of these values is knowledge added during the design and materialization process of the exchange identifiers database.

A given organization can be, of course, both a Supplier and a User, even for the same data item. The Supplier of an attribute name in the exchange identifiers database is the only one that can insert, modify, or delete values for that attribute in the database itself.

The active part of a system based on the AKW approach features two main components, each of them providing a class of services:

- *system services*, to keep coherence among the various sources of data items and of changes to them, to avoid incoherence during updates from Suppliers and their dispatching to Users, to certify answers to Users queries, to implement security and access right controls;
- *application services*, allowing Users to access data items they need, and allowing Suppliers to change data items or to generate new ones, both in a punctual way (i.e., one at a time) and in a batch one (i.e., a set at a time).

Application services are those specific for the cooperative information system supported by the AKW architecture. The set of databases affected by the cooperative information system is called the *application domain* of the AKW architecture. System services are generic and independent from the specific application domain, while application services are highly dependent on the domain requirements. There is a set of common functionalities in application services, for the management of the exchange identifier database, but this is a rather standard part. Hence in Section 7. we shall only discuss system services.

Note that a single cooperative information system based on the AKW approach may be designed so as to support more than one application domain. In such a case, a different set of application services is defined for each domain<sup>2</sup>.

#### 4.1. Advantages of our approach

With the AKW approach the following advantages are obtained:

- an organization is not deprived from a full control over its own data, which continue to be managed within its boundaries; hence this approach is “organization safe”;
- an organization has not to change its way of working or its computerized information system; in other words, the approach is “organization driven”, since the technical solution matches the organization’s behaviour;

---

<sup>2</sup>The issue of interrelations among many application domains and their cooperative information systems is at a higher level of complexity and is currently being analyzed in AIPA. Application domains sharing some data at conceptual level might also share some application services. But in this paper we focus only on the building of single cooperative information system.



- the system that is built can be managed by any of the organizations involved in the interaction, even if the access keys are not created/managed by the chosen organization. In fact, since the exchange identifiers database knows who is the owner of a key, it does not allow the control over that key to go outside the controlling organization. In other words, it knows that the only “certified instance” of the access key can be found at the organization owning it.

Please note that the last point above is really critical to obtain that organizations are really involved in the cooperative application development, since otherwise the desire of not losing control over data that are crucial for their mission will keep them away from cooperation.

From an economical point of view, our approach has three characteristics making it highly cost-effective:

- it makes it possible a quick start in the cooperative application development, since none of the existing systems has to be discontinued and/or changed;
- it makes it possible a cheap start, since the new portion of software to be developed is rather thin;
- performances obtained in applications developed with our approach are, on the end-user side, largely better than what can be obtained with encapsulation-based approaches.

Our approach also allows an incremental development of the technical solution, and this brings in two additional advantages:

- it reduces the complexity of the development process, by allowing to implement functions of the overall system in an incremental way;
- it allows a two-phase development, where:
  - in the first phase (*set-up*) the coherence manager functions are centralized, and this makes it possible to obtain a faster convergence to an initial level of coherence among the distributed data sources by means of a stricter and more centralized control;
  - in the second phase (*regime*) the cooperative application can be distributed and decentralized by bringing the coherence manager functions closer to data sources, thus decreasing the organizational overhead on the central offices.

Finally, an important technological issue has to be mentioned. Currently, one of the biggest difficulties in the development of cooperative applications distributed over Internet is the absence of an “Internet-oriented” transaction manager. Management and control of data flows exchanged by applications interacting over Internet is nowadays executed at the various nodes where applications reside. With the current technology, in the case of cooperative applications in the scenario we have described, there is not a real alternative since the only kind of feasible technical

solution (i.e., encapsulating these interactions in a shell environment controlling them) is organizationally unfeasible, due to the fact that it would subordinate all the applications to such a shell.

Therefore, whenever a new information system is implemented by means of the cooperation of applications running at different nodes over Internet, the best it can be done with the current technology is to check, control and record what happens at the various application nodes. But this makes it really hard and extraordinarily complex to verify that the interaction between various applications is developing as planned. It also makes it very difficult to reconstruct, in case of interaction problems, the temporal and causal sequence of steps and actions carried out. Until this technological issue is not really solved, it will not be possible to efficiently implement cooperative applications that are really effective.

The AKW approach we have introduced provides a leverage point for this important issue. In fact, it makes it available a single logical place, namely the exchange identifier database, where all interactions among the applications are naturally processed in an application-oriented way. In the AKW approach this can be executed without introducing subordination relations among the various applications and only for the exchanges of data that are really critical for the interaction among the applications. Hence it provides a very efficient framework for the development of a transaction manager for cooperative applications over Internet. How such a transaction manager can actually be implemented is a task we have not yet tackled, but we are very confident that our framework is an excellent starting point.

Furthermore, an important low-level data management requirement for such a development is the ability to control application threads at the network level. This in turn requires the ability to retrieve and classify network packets with high efficiency. We have developed a very effective solution to this requirement [10].

## 5. A formalization of the exchange identifiers database

For this formalization it is useful to make reference to the framework for a Data Warehouse architecture described in [6, 7]. For the sake of clarity, in the following we briefly describe such a framework.

The framework is characterized by the following elements:

- the *Conceptual Data Warehouse Scheme* representing the concepts that are of interest of the Data Warehouse application;
- the *Logical Data Warehouse Scheme* describing at the logical level, in terms of the Conceptual Data Warehouse Scheme, the content of the *Materialized View Store* of the Data Warehouse;
- the *Logical Source Schemes* describing at the logical level, in terms of the Conceptual Data Warehouse Scheme, the Data Warehouse views of the *Source Databases*;

The Data Warehouse application makes up the Materialized View Store by starting from actual data in the Source Databases by means of *wrappers* and *mediators*. The wrappers map the physical structure of data stores to the Logical Source Schemes.

The mediators act according to the Logical Data Warehouse Scheme and ask the wrappers for actual data.

In our approach we propose a Data Warehouse Scheme of a different nature. We call it *Access Key Scheme* (AKS), since it is a scheme describing only access keys for items in the various sources. AKS is the scheme of the exchange identifiers database introduced in the previous section.

From a formal point of view we can describe AKS as it follows.

Let  $U$  be the whole reality of interest for the cooperative information system one is designing. Any element in  $U$  has associated the value of some *features* (in general, a very large number of them). Examples of features are the given name of a person, the color of a car, the price of a book, and so on. Relations in the various Source Databases represent (elements of)  $U$  by storing values for various features of the elements of  $U$  as values of their attributes. Each Source Database of course represents the features that are more relevant to the fragment of the reality of interest it is modeling.

Features are specified by partial functions  $f : U \mapsto D_f$ , where  $D_f$  is a domain whose extent is invariant with respect to the state of the overall system. Values of  $D_f$  have to be representable in a logical data model for databases. Examples of  $D_f$  are: integers, char, strings, and so on. Hence, features are functions whose role is to provide values, stored in the database as values of attributes in tuples, to characteristics of elements of  $U$ .

AKS is a set of relation schemes defined so as to allow to support coherence maintenance among a set  $\mathcal{A}$  of attributes belonging to Source Databases. In order to specify AKS, for each attribute  $a \in \mathcal{A}$  of a relation  $R$  in a Source Database:

- a uniquely identified feature name  $f_a$  has to be given, and
- a function  $i_a : R \mapsto U$ , called *reference* for  $a$ , has to be specified; the reference gives an interpretation of  $R$  in terms of the elements of  $U$ : it gives, for each element  $t$  of  $R$ , the element  $i_a(t)$  of  $U$  having  $t.a$  as value of the feature  $f_a$ ; and
- a declaration to be of kind either *supplier* or *user* has to be provided; correspondingly, we say that  $R$  is either a *supplier* or a *user* of  $f_a$ .

The reference  $i_a$  is completely specified:

- by selecting a subset  $I_a$  of attributes of  $R$  such that for each  $t_1, t_2 \in R$  it is  $i_a(t_1) = i_a(t_2) \Leftrightarrow I_a(t_1) = I_a(t_2)$ , where  $I_a(t) = \Pi_{I_a}(t)$ ; and
- by knowing the mapping  $m_{I_a} : I_a \mapsto U$ .

With the above, for each  $t \in R$ , we have that  $i_a(t) = m_{I_a}(I_a(t))$ . Please notice that, in general,  $I_a$  is not an index for  $R$ .

With the above positions, an overall representation is defined for any feature name  $f_a$  such that  $a$  is of kind supplier. This representation is defined by the aggregation of values of attributes of the kind supplier sharing the feature's name, according to the following definition.

**Definition 5..1** For each  $x \in U$  the value of  $f_a(x)$  is defined if and only if for each tuple  $t$  in each relation  $R$  in any Source Database a unique value  $y$  exists such that the fact that  $R$  is a supplier of  $f_a$  and  $i_a(t) = x$  implies  $t.a = y$ . Then the value of  $f_a(x)$  is  $y$ .

We can now formally define what is the coherence of a relation.

**Definition 5..2** A relation  $R$  is coherent with respect to the cooperative information system if and only if for each attribute  $a$  of  $R$  which is of kind user and for each  $t \in R$  it is  $f_a(i_a(t)) = t.a$ .

Then AKS is a set of relation schemes such that, for each attribute  $a$  whose coherence needs to be maintained in the cooperative information system, a relation scheme  $S_a$  exists in AKS such that, for each relation  $R$  in any Source Database which is a user of  $a$ , attributes of  $I_{R.a}$  representing reference  $i_a$  belong to  $S_a$  and are an index for it. In this way,  $S_a$  surrogates  $U$  with respect to feature  $f_a$  by means of the representation of  $m_{I_a}$  in terms of values of attributes  $I_a$ . Note that, given two relations  $R$  and  $Q$  and an attribute  $a$ , belonging to both of them, for which  $R$  is a supplier and  $Q$  is a user, then  $I_{R.a}$  and  $I_{Q.a}$  need not to be the same list of attributes.

For example,  $R$  is { name, birth-date, birth-place, address },  $a$  is address, and  $Q$  is { name, social-security-number, address },  $I_{R.address} = \{ \text{name, birth-date, birth-place} \}$ , and  $I_{Q.address} = \{ \text{social-security-number} \}$ . Then, to maintain the coherence of the values of address, in AKS we have  $S_{address} = \{ \text{name, birth-date, birth-place, social-security-number} \}$ .

Note that the correctness of an attribute value is not something directly dealt with in the definition and materialization of the AKS. Suppliers for a given attribute are the only ones responsible for correctness, while AKS is only used in dealing with the maintenance of coherence among relations supplying values and relations using them.

An AKS requires, in general, that mediators and wrappers are of a different nature than in the standard framework. Also, having an AKS implies, from an extensional point of view, that the content of an Access Keys Warehouse is always a subset of a Data Warehouse for the same Enterprise Model. But anyhow, the choice of having an AKS instead of a full Data Warehouse Schema, while allowing to fully satisfy requirements imposed by the scenario described in Section 2. is not forbidding to extend the Access Key Warehouse to a fully blown Data Warehouse.

## 6. Some typical scenarios and the use of AKW approach

In this section we discuss three typical scenarios of the use of the AKW approach. Cases are presented with reference to the framework of the Conceptual Data Warehouse Scheme [6, 7] recalled in Section 5..

Note that the materialization of the exchange identifiers database requires knowledge that is not necessarily present in any of the Source Database and has to be

added during the materialization process (see also the first scenario below). Note that such a knowledge is extensional since it allows to say that two elements  $t$  and  $t'$  in  $U$ , retrieved by means of two, generally independent, indexes  $I_a$  and  $I'_a$ , applied to two different relations, are the same element of  $U$ . Hence no change to any schema in a Source Database is required or implied.

As suggested in Section 7., to make the execution of Synchronization services more efficient the index for attributes can be enlarged beyond what is strictly necessary to uniquely identify an element of  $U$  (see also the second and third scenario below). This is a crucial design decision and the AKW approach offers a clear and precise framework to deal with it.

### 6.1. Synchronization between entities

Let  $R_1$  and  $R_2$  be two relations in two different Source Databases. Assume  $R_1$  and  $R_2$  represent respectively two classes  $C_1$  and  $C_2$  in their Source Schemes and  $C_1$  and  $C_2$  are both subclasses of a same class  $C$  of the Conceptual Data Warehouse Scheme. Assume also  $C$  has a feature  $\Phi$  (see figure 1). For example,  $C_1$  is the class of the residents in a certain city,  $C_2$  the class of the tax-payers,  $C$  the class of the persons, and  $\Phi$  is the birth-date.

Assume  $R_1.a_1$  and  $R_2.a_2$  have associated the same feature name  $\Phi$ , where  $R_1.a_1$  is a supplier with index  $I_{R_1.a_1}$ ,  $R_2.a_2$  is a user with index  $I_{R_2.a_2}$ , and  $I_{R_1.a_1}$  and  $I_{R_2.a_2}$  are superkeys for, respectively,  $R_1$  and  $R_2$ . Values provided by the superkeys provide, in general, different representation for a same element of  $U$ . The (extensional) knowledge represented within  $R_1$  and  $R_2$  does not allow, in general, to keep the coherence among values of  $R_1.a_1$  and  $R_2.a_2$ .

This knowledge is expressed in the materialization of AKS using  $I_{R_1}$  and  $I_{R_2}$ . Such a database hence associates superkeys of relations involved so that it is possible to correctly determine, for each  $t \in R_1$  and for each  $t' \in R_2$  if  $i_{R_1.a_1}(t) = i_{R_2.a_2}(t')$  or not.

### 6.2. Synchronization between an entity and a relation

Let  $R_1$  and  $R_2$  be two relations in two different Source Databases. Assume  $R_1$  and  $R_2$  represent respectively two classes  $C_1$  and  $C_2$  in their Source Schemes. Class  $C_2$  represents the association of elements of  $C_1$  with elements of a third class, in the sense that each element of  $C_2$  is a couple made up by one element of  $C_1$  and one element of the third class. Assume also  $C_1$  has a feature  $\Phi$  (see figure 2). For example,  $C_1$  is the class of the residents in a certain city,  $C_2$  the class representing the ownership, by a resident, of a building, and  $\Phi$  is the birth-date.

Assume  $R_1.a_1$  and  $R_2.a_2$  have associated the same feature name  $\Phi$ , where  $R_1.a_1$  is a supplier with index  $I_{R_1.a_1}$  and  $R_2.a_2$  is a user with index  $I_{R_2.a_2}$ . In this case  $I_{R_1.a_1}$  is a superkey for  $R_1$ , while  $I_{R_2.a_2}$  may not necessarily be a superkey of  $R_2$ . In fact, each element of  $R_2$  may be the aggregation of many components of which the feature with name  $\Phi$  characterizes only a part. E.g., an instance of  $R_2$  may represent the sale act of the building.

In such a case the enlargement of  $I_{R_2.a_2}$  with other attributes regards only the efficiency of retrieval in  $R_2$  of all tuples where a variation of the value of  $a_1$  in  $R_1$

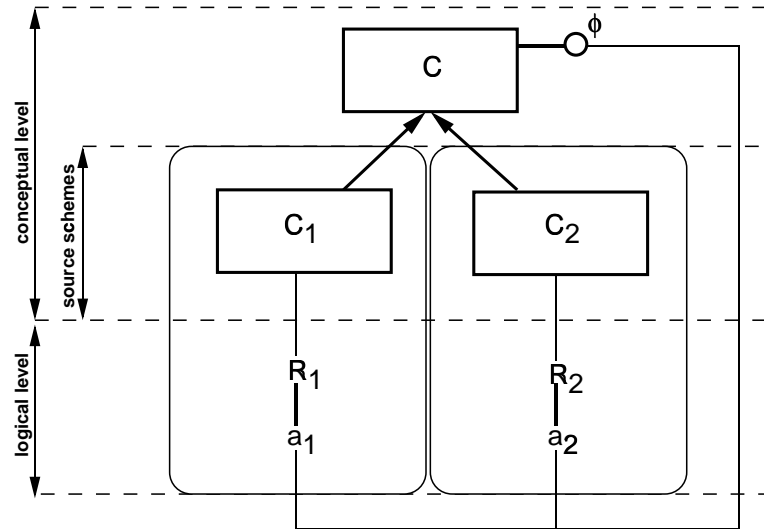


Figure 1: Synchronization between entities

has to be reported.

### 6.3. Synchronization between an aggregation class and its components

Let  $R_1, \dots, R_n$  and  $R$  be  $n+1$  relations in  $n+1$  different Source Databases. Assume  $R_1, \dots, R_n$  and  $R$  represent respectively  $n+1$  classes  $C_1, \dots, C_n$ , and  $C$  in their Source Schemes, where  $C_1, \dots, C_n$  are a partition of  $C$ . Assume also  $C$  has a feature  $\Phi$  (see figure 3). For example,  $C_i$ , for  $1 \leq i \leq n$ , represents the residents in a given municipality, while  $C$  represents the residents in the whole country. This is subject to the constraint that each resident in the country has to belong to exactly one municipality and vice-versa each resident in a municipality is a resident in the country. Feature  $\Phi$  represents the birth-date.

Such a scenario has two variations, corresponding to two different organizational realities. They are characterized by the fact that  $R$  is a supplier or a user of  $\Phi$ .

Assume then, in the first case, that  $R_i.a_i$ ,  $1 \leq i \leq n$ , and  $R.a$  have associated the same feature name  $\Phi$ , where  $R_i.a_i$  is a supplier with index  $I_{R_i.a_i}$  and  $R.a$  is a user with index  $I_{R.a}$ . In such a case Definition 5.1 causes the defeating of values for attributes  $a_i$  and  $a_j$  for every element of  $U$  represented in relations  $R_i$  and  $R_j$ ,  $i \neq j$ , with different values for  $a_i$  and  $a_j$ . In this way the system would provide different values for the same feature name: this is equivalent to say that this value is unknown, hence it cannot be reported to users of it. The coherence manager introduced in Section 4. uses this defeating mechanism to avoid different Source Databases provide different values for the same element of  $U$  and also to enforce in Source Databases constraints like the partitioning one introduced above.

In the second case the assumption is that  $R_i.a_i$ ,  $1 \leq i \leq n$ , and  $R.a$  have

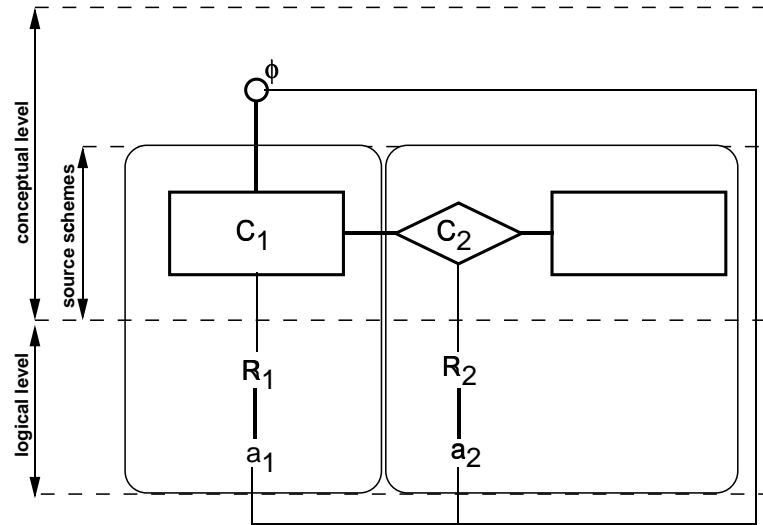


Figure 2: Synchronization between an entity and a relation

associated the same feature name  $\Phi$ , where  $R_i.a_i$  is a user with index  $I_{R_i.a_i}$  and  $R.a$  is a supplier with index  $I_{R.a}$ . Now, under the further assumption that residents are uniquely identified only if the name of municipality where they are resident is also used, to allow for a correct synchronization of variations of values of the birth-date index  $I_{R.a}$  has to be enlarged to allow the identification of the relation corresponding to the municipality of residence. On the other side, if residents have a unique identification key independently from the municipality of residence, the enlargement of  $I_{R.a}$  is just a matter of efficiency of the system for identifying the relation interested by a variation.

## 7. Dynamics of an AKW system

We now give a conceptual description of the **system services** of the AKW approach, that are the dynamic part supporting coherence maintenance functions. We do not describe in this paper services supporting AKS design activities.

In the following, when we speak of interaction between an organization and the AKW system, of course we mean the interaction between the computerized information system of the organization and the AKW system.

Communication between the AKW system (i.e., a cooperative information system based on the AKW approach) and Source Databases is based on the **asynchronous model**. This has been chosen for reasons of effectiveness, reliability, and security:

- it allows an easier inter-connection of heterogeneous systems;
- it allows a better decoupling between Users and Suppliers, also from an organizational point of view;

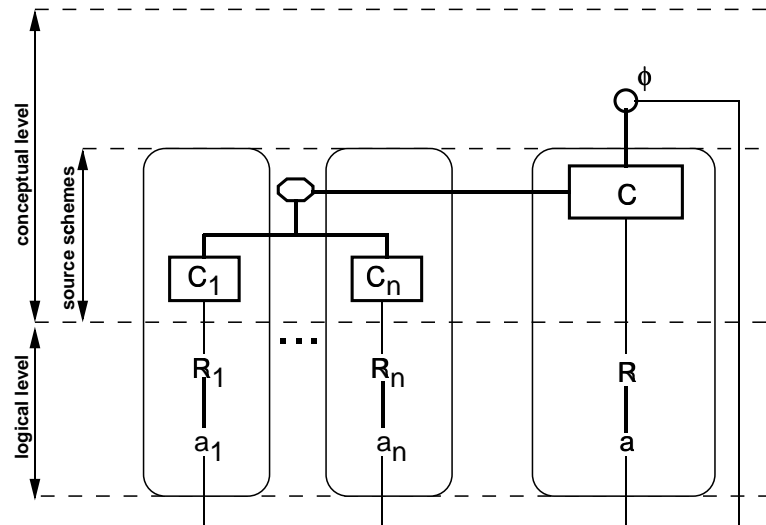


Figure 3: Synchronization between an aggregation class and its components

- it provides a better support to certification activities;
- it can be easily extended to support ‘on-line’ access;
- it agrees with the trends for technological solutions in the field of cooperative applications.

There are five classes of system services:

- *Certification*: services in this class check that the interaction among Users and Suppliers has developed as planned and certify it on demand.
- *Publication*: services allowing to let an organization to declare its attribute(s) as public and to allow organizations to know which attributes have been made public by others. Both supplier and user attributes can be made public. In the former case, this allows to other organizations only to query its values and/or to receive its updates, while in the latter one this means that other organizations can request to the organization that has made the attribute public to change its values.

Note that only an attribute  $a$  for which a relation scheme  $S_a$  has been defined in AKS can be made public, otherwise the AKW system would not be able to support coherence maintenance for them.

- *Acceptance*: services allowing to an organization that is interested to variations of an attribute  $a$  for which a relation scheme  $S_a$  has been defined in AKS to let the AKW system knows this interest, so that any future change of attribute values is also forwarded to the organization, together with values for attributes in  $I_a$ .



To this request, a set of other attributes in AKS can also be associated with two purposes:

- to obtain the updated value together with a set of correlated values that are relevant for the organization to efficiently maintain internal coherence in its Source Database,
  - to enable the AKW system to dispatch the updated value only on the satisfaction of suitable conditions on the correlated values so that the overall performances are not degraded.
- *Synchronization*: whenever a supplier attribute made public in AKS changes its value, services in this class allow to the AKW system to receive the update from the supplier and to dispatch it to other organizations having requested to be informed. To enable the notification of the change, correlated values are retrieved and the satisfaction of conditions is checked.

Note that the introduction of the exchange identifiers database is critical to implement Synchronization services without the unacceptable degradation of performances of the overall system that would derive from an unrestricted broadcast of any change to all organizations.

- *Alert*: managing incoherence signals. It may happen that a change in an attribute value, when notified to a requesting organization, raises a local coherency problem between its new value and the current situation existing in Source Database of the notified organization. In such a case the notified organization will certainly not update its Source Database, but will send an incoherence signal to the AKW system.

Note that the AKW system will not usually take a decision about how to solve this incoherence, since this is an organizational problem. AKW only offers an efficient technical solution to be aware of and to deal with this incoherence.

Reasons of effectiveness, reliability, and security have led to the decision to design all data flows so as to be fully **self-identifiable**. This means that all data needed to completely identify source and destination of the flow, service requested and its outcome, service and quality parameters, are contained within the flow itself.

This allows to the AKW system:

- a lower complexity and a greater flexibility in the management of flows (i.e., dealing with routing, tracing, security and certification);
- a better transparency, from the end-user point of view, with respect to the physical addresses of Users and Suppliers;
- a higher security in the interaction among Users and Suppliers.

All data flows in the AKW system are also tagged with the relevant service parameters (e.g. transaction date and time, security level, authorization, etc.). Hence service parameters of a data flow are not discussed in the following.

We now describe in details the **system services**, that are the more interesting part of the AKW architecture. A graphical description in terms of Data Flow Diagrams is presented in figure 4.

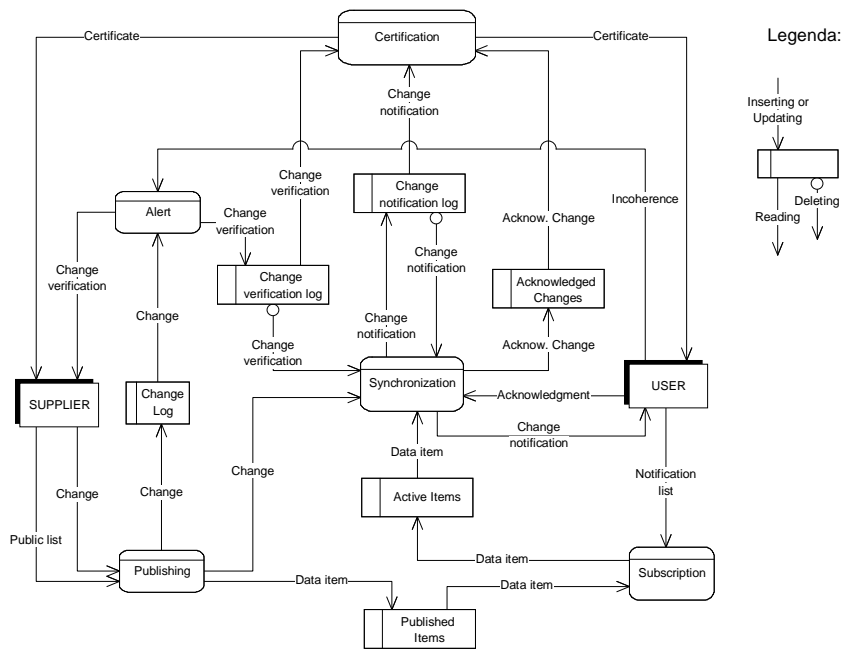


Figure 4: System services in the Access Keys Warehouse approach

### 7.1. Certification: monitoring user-supplier data flows

Services in this class check that the interaction among Users and Suppliers has developed as planned. They also certify this interaction and record it for any future certification request. Hence there are services to manage:

- the request of a specific application service, characterized by:
  - identifier of the user requiring the service,
  - identifier of the supplier the user addressed the request to,
  - application service requested (e.g., querying, deleting, changing, etc.) and its input parameters,
  - identifiers of data items and databases affected by the request,
- the outcome of a specific request, characterized by:
  - identifier of the supplier that serviced the request,
  - state/outcome parameters of the request (e.g., delayed, being processed, rejected, confirmed, aborted, etc.),
  - databases affected by the request,
- coherency parameters for the outcome of a request, characterized by:

- identifier of the user requiring the service,
- identifier of the supplier the user addressed the request to,
- application service requested (e.g., querying, deleting, changing, etc.) and its input parameters,
- identifiers of data items and databases where the user detected coherency problems as a consequence of the requested service,

### 7.2. Publication: *accepting requests of distributing changes*

In this class there are services allowing to an organization that is willing to share changes to its databases to let the AKW system know which are the data items it can make available.

The organization hence communicates to the AKW system a list of identifiers of data items (called *public list*) whose changes are public. In such a way, other organizations interacting in the application domain can be advised, if they have done a request for a change notification signal for an item in this public list, of any update occurring to the item itself.

Whenever a data item which is in the public list is changed, services of the AKW system in this class are advised of the variation, receive it and store all relevant details about it in the *change log file*. Organizations interested in such a change are then automatically advised if they have subscribed to the change notification service (see services in the next two subsections). A change is logically<sup>3</sup> removed from the change log file only when all notifications of changes have been acknowledged from all subscribed organizations.

### 7.3. Acceptance: *managing request for change notification signals*

In this class there are services allowing to an organization that is interested to variations to a data item to let the AKW system know this interest, so that any future change of the data item is also forwarded to it.

An organization communicates to the AKW system a list of identifiers of data items (called *change notification list*) it is interested to know the changes of. Corresponding to each data item in the change notification list, there is a *data schema*, describing the structure that has to have the information about the change that has to be given back to the requesting organization. This data schema is filled by the AKW system by using data items from databases in the application domain and according to a set of *correlation rules* that has also been provided by the requesting organization for each data item in the change notification list.

### 7.4. Synchronization: *receiving and dispatching updates*

Whenever a data item in the application domain, whose change-ability was made public by the relevant organization, is modified, services in this class receive information on the change by the Publishing services (see subsection 7.2.) and any organization having in its change notification list the identifier of this data item is automatically advised.

---

<sup>3</sup>From a physical point of view it remains there for administrative purposes.

Hence there are services to:

- receive a change message, characterized by:
  - identifier of the supplier generating the change,
  - valid date and valid time for the change,
  - type of operation (e.g., insert, update, delete, etc.),
- build and deliver a change notification message, characterized by:
  - identifier of the supplier generating the change,
  - valid date and valid time for the change,
  - type of operation (e.g., insert, update, delete, etc.),
  - data schema requested for the change notification, filled with data items related to the changed item,
- manage the *change notification log file*, containing a list of all change notification messages not yet acknowledged by organizations that have subscribed to the change notification service for a given data item. Whenever an organization has received the change and has executed all the relevant variations to its databases without detecting any coherence problem, by acknowledging the notification it certifies to the AKW system the compliance of its information system with the new situation determined by the change. On receiving this acknowledgment, or a new change when an incoherence signal has been issued, the related change notification message is logically<sup>4</sup> deleted from the change notification log file;
- update the *change verification log file*, whenever the change received is related to a previous change that caused an incoherence signal (see also subsection 7.5.).
- manage the *acknowledged change log file*, storing a list of all changes that have been eventually acknowledged by all subscribing organizations; a record inserted here certifies that the overall set of data has moved from one coherent state to another coherent state.

#### 7.5. Alert: managing incoherence signals

It may happen that a change in a data item, when notified to a subscribing organization, raises a coherency problem between its new value and the current situation existing in databases internal to the notified organization. In such a case the notified organization will certainly not update its databases, but will send an incoherence signal to the AKW system.

Service in this class hence are able to:

- receive an incoherence signal, characterized by:
  - identifier of the supplier that generated the change causing the incoherence,

---

<sup>4</sup>See previous note.

- valid date and valid time for the change causing the incoherence,
  - type of change operation (e.g., insert, update, delete, etc.),
  - identifier of the organization that subscribed to the change notification service for this data item and detected the incoherence,
  - data schema requested for the change notification, filled with data items related to the changed item,
  - identifiers of data items and databases where the notified organization detected coherency problems as a consequence of the change notification,
- correlate an incoherence signal with the relevant change in the change log file and build a *change verification message*: note that in the case of multiple incoherence signals, AKW system tries to gather them in a single change verification message, using a ‘time-out’ approach while waiting for acknowledgements of change notification from subscribed organizations;
  - send a change verification message to the supplier that has generated the change which has to be verified and record it in the *change verification log file*: the supplier has to anyhow answer with a new change that is processed by the Synchronization services (subsection 7.4.) of the AKW system. On receiving this new change the change verification message is logically<sup>5</sup> removed from the change verification log file.

## 8. Conclusions

In this paper we have introduced a novel architectural approach, namely the **Access Keys Warehouse** approach, to be used for the development of cooperative information systems supporting the integration among legacy information systems of autonomous organizations.

This approach proposes a novel role for the concept of Data Warehouse, namely suggests that a (new kind of) warehouse can be set up to guide and control accesses to underlying databases. This allows to solve coherency problems with good overall performances and to provide a methodological guidance for the development of a cooperative information system.

It also lays the foundation for the development of an “Internet-oriented” transaction manager, enabling an efficient and effective control over inter-organizational cooperative applications interacting over Internet.

From the organizational standpoint the AKW approach has following advantages:

- it is “organization safe”, since leaves to each involved organization the full control over its own data and their changes;
- it is “organization driven”, since the technical solution matches organization’s behaviour.
- it lowers the barriers keeping organizations away from the cooperative application development.

---

<sup>5</sup>See previous footnote

## References

- [1] M.Tamer Ozsu, P.Valduriez, *Principles of Distributed Database Systems*, Prentice Hall Canada, 1999, 2nd edition.
- [2] G.Zhou, R.Hull, R.King, Generating data integration mediators that use materializations, *Journal of Intelligent Information Systems*, 6, pp.199–221, 1996.
- [3] J.Widom, Research Problems in Data Warehousing, 4th International Conference on Information and Knowledge Management (CIKM'95),
- [4] AIPA, Feasibility Study of the Public Administration's Unified Network, January 1996, [http://www.aipa.it/english/unifiednetwork\[2/feasibilitystudy\[1/index.asp](http://www.aipa.it/english/unifiednetwork[2/feasibilitystudy[1/index.asp).
- [5] A. Umar, *Object Oriented Client/Server Internet Environments*, Prentice Hall, 1997.
- [6] D.Calvanese, G.De Giacomo, M.Lenzerini, D.Nardi, R.Rosati, Information integration: conceptual modeling and reasoning support, 6th International Conference on Cooperative Information Systems (CoopIS-98), pp.280–291, 1998.
- [7] D.Calvanese, G.De Giacomo, M.Lenzerini, D.Nardi, R.Rosati, Source Integration in Data Warehousing, 9th International Workshop on Database and Expert Systems Applications (DEXA-98), pp.192–197, IEEE Computer Society Press, 1998.
- [8] Law n.133, 26 february 1994 and Law Decree n.557, 30 december 1993, art.9 (in italian).
- [9] M.Talamo, F.Arcieri, G.Conia, Il Sistema di Interscambio Catasto-Comuni (parte I), *GEO Media*, vol.2, Jul-Aug 1998, (parte II), *GEO Media*, vol.2, Sep-Oct 1998, Maggioli Editore, Roma (in italian).
- [10] E.Nardelli, M.Talamo, and P.Vocca. Efficient searching for multidimensional data made simple. 7th Annual European Symposium on Algorithms (ESA'99), Prague, Czech Republic, Jul.99, Lecture Notes in Computer Science vol.1643, pp. 339–353, Springer-Verlag.
- [11] Il Catasto Telematico, *Notiziario Fiscale*, n.11-12, pp.19–22, Nov-Dec 1998, Ministry of Finance, Roma (in italian).
- [12] M.Talamo, F.Arcieri, G.Conia, E.Nardelli, SICC: An Exchange System for Cadastral Information, 6th International Symposium on Large Spatial Databases (SSD'99), Hong Kong, China, Jul.99, Lecture Notes in Computer Science vol.1651, pp.360–364, Springer-Verlag.
- [13] F.Arcieri, C.Cammino, E.Nardelli, M.Talamo, A.Venza, The Italian Cadastral Information System: a Real-Life Spatio-Temporal DBMS, Workshop on Spatio-Temporal Database Management (STDBM'99), Edinburgh, Scotland, Sep.99, Lecture Notes in Computer Science, vol.1678, pp.79–99, Springer-Verlag.

- [14] F.Arcieri, E.Cappadozzi, P.Naggar, E.Nardelli, M.Talamo, Specification and architecture of an Access Key Warehouse, Technical Report n.7/99, Univ. of L'Aquila, March 1999.

## A The Case of the Italian Cadaster

Here we describe a real-life example of the scenario introduced in Section 2., taken from the Italian PA.

### A1. Cadastral Data in Italy

The Cadaster, in the italian situation, has the role of being the public registry of real estates and land properties. As such, it has been always managed at the central administration level, namely by the italian Ministry of Finance. The access key in the Cadaster to data about real estates and land properties is expressed in terms of the unique code of municipality where they are located and of four cadastral codes referring to cadastral maps of increasing level of detail.

A Municipality has the objective of planning and managing land use. For this purpose it mainly uses toponymy information about properties. Hence access key for Municipalities is street name, plus possibly house number on the street and flat number.

From a physical point of view, Cadaster data are managed by the Land Department of the Ministry of Finance through its Land Offices (“Uffici del Territorio” = UTEs) that are present at the level of Provinces, which are a subdivision of the main administrative partition of Italy in Regions and an aggregation of Municipalities.

The Ministry of Finance, as required by the law, uses cadastral data to keep record of and to certify location and planimetry of properties. Note that, according to italian law, taxes on real estates and land properties have to be based on their cadastral value (“rendita catastale”), that is strictly depending on location and planimetry of properties.

Furthermore, through its Estate Public Registry Offices (“Conservatorie Immobiliari”), the Ministry of Finance also keeps record of and certify ownership rights and mortgage rights relative to properties.

Municipalities also have their databases about real estates and land properties. These are used, as set by the law, to support and manage actions in the sectors of Toponymy, Fiscality, Public Works, and Land Management.

It is hence clear that there is a continuous exchange flow of cadastral data among Municipalities, Ministry of Finance, Notaries, and Certified Land Surveyors.

Note also that cadastral databases are not managed at a single central location but at the more than 100 Land Offices (UTEs) of the Ministry of the Finance. This means that there is not a single centralized system, but more than 100 systems, geographically distributed over the whole italian territory.

Size of data bases managed by Municipalities is largely variable, considering that about 6.000 of the 8.102 italian municipalities have less then 5.000 citizens, but 8 of the 20 Region chief towns have more than one million inhabitants.

Typical queries on cadastral data bases are:

- *cadastral certification* query (“Visura Catastale”), requiring a certificate about location and cadastral value of a real estate/ land parcel; please note that such



a certificate is needed by notaries in all sale acts and buyers pay a fee to obtain it from the Cadaster;

- *planimetry certification* query, requiring a certificate about planimetry of a real estate; such a certificate is often required during sale transactions to check if the current situation of the real estate is coherent with respect to the situation recorded in the cadastral databases,
- *update* query, submitting a request to change, for a given real estate/ land parcel, some piece of information of geometric nature or of descriptive nature.

In one of the largest cities, every year there are about 750.000 requests for cadastral certificates.

The yearly number of geometric updates to cadastral databases is about 250.000. These updates always triggers further updates, since a geometric change affects one or more of the following aspects of a real estate or land property:

- property rights and mortgage rights,
- fiscal nature,
- destination and allowable usage.

To deal with coherence maintenance issues in cadastral data exchange, as required by the law [8], AIPA started in 1995 the SICC project [9, 11, 12, 13], with the participation of Ministry of Finance and ANCI, the association of italian municipalities.

#### *A2. How incoherence is generated in this case*

We now describe a typical interaction among entities that interact in the case of Cadaster to show how the generic example presented in Section 2. looks like in this specific case.

A Certified Land Surveyor (entity *A* of the generic example) prepares for a client a request for a variation to an apartment (e.g., to divide a large apartment in two smaller ones). The request is composed by some descriptive data and some geometric data and is stored in a database in the surveyor's office.

The surveyor prints the request and send it by registered mail to the pertinent cadastral office of the Ministry of Finance (entity *B*). The office, having checked that everything has been done according to current laws and that data are coherent with data stored in cadastral databases, executes the update.

The municipality the apartment is located in (entity *C*) has an interest in knowing such a change for local tax reasons (e.g., the two smaller apartments are different subjects, from a fiscal point of view, than the previous one). The surveyor has an obligation to get an approval for the change from the Building Service of the municipality before submitting the request to the Cadaster. Of course, until the request is received from the Cadaster the change has not really happened.

But neither the cadastral office nor the surveyor have any legal obligation to inform the municipality when the change really happens, i.e. when the request has

been accepted by the Cadaster. This is the duty of the owner of the apartment and if he/she forgets to comply with this obligation, the municipalities may never be aware of the change until an inspector is sent in the apartment to check the situation.

Let us now consider the coherence problem at the second level. Assume that, in the division of the apartment in two smaller ones, one of the two has received a new apartment number and has been recorded in the cadastral data base with it. Assume now the municipality is informed of the change, when it happens, but it discovers that the way the apartment has received the new number is incoherent with municipal regulations for numbering apartments (e.g., the new number is one plus the old highest number in the building while current regulations require adding a letter to the old number). Note that such a mistake may have been unnoticed or unchecked in the prior request for approval submitted from the surveyor to the municipality. In fact, the Building Service of the municipality is not the one in charge of such a check on apartment numbering (the Toponymy Service is in charge) and regulations require that the submission of the change request to the Cadaster only needs the approval of the Building Service.

When the municipality receives the communication of the change it will try to have the Surveyor and the Cadaster change their databases according to such a regulation. But since most probably cadastral databases will have been already updated by then and since this issue of apartment numbering is not something the Cadaster has, by the law, to really care about, no action will be taken and the incoherence will remain there.

### *A3. Implementation of the AKW architecture in the Cadaster case*

The use of the AKW approach fully supports the SICC project targets, since it allows to progressively synchronize the various distributed databases. This increase in database correlation means that data manipulation can be more and more decentralized towards municipalities while keeping, as required by the law, a central high-level control.

The first prototype of the SICC project was implemented in 1995 by AIPA and the Italian National Research Council. This prototype proved the feasibility of the technical solution and of the organizational model proposed.

Then SOGEI, the Italian company managing the computerized information system of the Ministry of Finance, developed a second prototype, with a better degree of integration among cadastral data and services. This prototype was put into operations in peripheral offices of Naples municipality in May 1997.

It was then subsequently validated, through the involvement of about 100 Municipalities ranging from Region chief towns to very small ones and a small sample of notaries and certified land surveyors, for about one year.

Finally, in September 1998 the engineered system, named SISTER [11] and developed as well by SOGEI, was put into nation-wide operation.

Access to the system is through a WEB-based interface and the effectiveness of its use is demonstrated by the sharp increase of requests managed by it during the

first months. In the month of January 1999 there were already more than 100.000 cadastral certification queries. Remember that such a query is usually paid by its final user.

The final phase of the whole project is running in 1999 and aims at extending the range of services provided to end users.