Interoperability and Workflow: Multi-Agency Databases

Rudolf N. Müller and Andrew U. Frank
Department of Geoinformation
Technical University of Vienna
{mueller, frank}@geoinfo.tuwien.ac.at

Abstract

Many agencies hold substantive data collections. The cost of collecting the data is high, and the cost of maintaining the data is even higher. There is an economic incentive to share the data with other agencies. These agencies cooperate with other agencies and the sharing of data facilitates the business processes. These vast collections of data are managed by federated database systems. The drawback of such systems is that they are insufficient to take care of the different authority structures in the participating agencies. We propose an extended workflow, which uses storable transaction scripts, so-called Update Proposals, and dedicated user roles to authorise proposed changes of the underlying federated database by Update Proposals. These Update Proposals are written in a vendor-independent Transaction and Transformation Language. The storable transaction scripts also implement a generalised form of "long transactions".

1 Introduction

Many agencies have acquired substantive data collections over the last decades. In these agencies data collection and maintenance is typically expensive. It is economically advantageous to share the data with other agencies without unnecessary replication. Therefore agencies provide other agencies access to their data and access data from other agencies. This is the standard assumption of federated databases, and it becomes increasingly real with today's network technology of data sharing among independent agencies.

An European project (COMMUTER) has studied the requirements in three towns (Lille, Harburg and Bologna) and found that sharing data in a federated database environment is hindered by the autonomy of each agency to perform updates: only few officers within the agency have permission to update records. This is a legal requirement, and we must provide technical means to build systems that fulfil this requirement. Other agencies can access the data, but they must not change it. Updates by non-authorised agents are sometimes necessary, sometimes very desirable. An example is: An agency asks an outside company to prepare a complex change (e.g., an engineering project), but this outside company cannot legally change the database. The problem is to achieve optimal workflow in this situation of cooperative work, regulated by legal roles of authority.

A possible solution for this problem is the use of federated database systems (FDBS) extended by a flexible transaction mechanism, which respects the update authorities. These database systems provide access to a set of pre-existing local databases management systems (LDBMS). The main features of a FDBS are heterogeneity and autonomy. Autonomy dictates that a LDBMS participating in a FDBS should not be modified by the FDBS and has the right to decide which types of internal information can be provided to the FDBS and to execute queries and transactions according to its own rules [8]. Due to autonomy there are two types of transactions in the entire FDBS. These are local transactions within a LDBMS and global transactions. Local transactions are executed by the LDBMS without the control of the FDBS. A global transaction, which accesses data in more than one LDBMS, is submitted to the FDBS. There it is decomposed into several sub-transactions and executed by the different LDBMSs. Substantive research has been conducted to resolve the problems that arise with global transactions, like low concurrency, possibility of global deadlocks or wasteful resource consumption. Refer to Hwang [5], who proposes a very interesting solution to the problem of concurrency control in a FDBS.

This paper, however, deals with the problem that the agencies that maintain LDBMSs participating in a FDBS are often restricted by rules defining update authorities. Update authority in this context means that there may be organisational restrictions to who may have the right to update other LDBMSs participating in the FDBS. Typically, one or a few officers of an agency have to authorise all updates to the database of the agency. How an FDBS-system can be built around these restrictions is the topic of this paper. It is concerned with interoperability of tasks and workflow.

Legal restrictions regarding update authority can be dealt with by splitting a transaction in two: in a first phase an Update Proposal is prepared, which contains instructions of what has to be changed. The authorising officer of the agency then checks this Update Proposal and he authorises

Müller, R., and A.U. Frank. "Interoperability and Workflow: Multi-Agency Databases." Paper presented at the 6th Int. Conference on Distributed Multimedia Systems (DMS99), University of Aizu, Japan 1999.

the execution of this update, which is then performed as a database transaction. This principle is more general and can be used to solve the problem of so-called "long-running transactions", i.e., transactions that will not be executed immediately after creation, but remain uncommitted in the system for a longer period of time [6, 1]. Update Proposals are a way to extend FDBSs by means of authorised updates to the FDBSs and to provide a way to handle long transactions.
This paper is organised as follows. Section 2 gives several motivating examples of the problem. Section 3 describes the user roles in this global environment. Section 4 contains the description of the architecture of the system. Section 5 deals with the workflow of Update Proposals. Section 6 describes the application of the workflow to the examples given in section 2. Section 7 discusses the aspect of long-running transactions in the proposed system. Section 8 deals with the Transformation and Transaction Management Language (TTML), which is used to provide system-independence. Section 9 describes the use of metadata in the proposed system on principle.

2 Motivating Examples

This section gives several examples of situations typically encountered when dealing with a multi-agency environment [11]. Please note that this list of examples is not exhaustive and many more similar examples could be found.

2.1 Changes by Outside Agents

The situation addressed here is the distribution of an update between two agents, one without, one with the authority to update the database. This can occur in many situations and an example from public administration serves to demonstrate the general problem:
An engineering company is preparing a plan for the construction of a new street. The company prepares changes to the cadastral database – small pieces of land are taken away from adjoining parcels and added to the public road. The engineering company does not have the authority to change the FDBS, and submits the plans for approval to the concerned authority. The cadastral authority checks the legality of the proposed changes and after approval inserts the changes in the database. Today this requires a redigitizing of the submitted plans, which is slow and error prone.

2.2 Conditional Validation

Update Proposals often rely on other Update Proposal as pre-conditions.
For example: an outside agent submits plans for a new house. The prospective landlord only recently bought the parcel on which the house will be located. So the Update Proposal containing the plans for the new house only may be accepted into the FDBS if, and only if, an Update Proposal changing the ownership of the concerned parcel was successfully accepted into the FDBS.

2.3 Correction of Out-Dated or Incorrect Data

Often Update Proposal Makers encounter incorrect or outdated data in the FDBS. This data has to be updated not only to allow an Update Proposal to be accepted into the FDBS but to keep the FDBS as up-to-date as possible.
Now these corrections are often done on an informal basis, as there are often no procedures defined to correct these entries.

2.4 One Proposal Changes Several Databases

One of the most costly operations is to update data which is either duplicated or distributed throughout the FDBS, i.e. the entries of interest are either held at several LDBMSs to make access to these entries easier, or several semantically connected entries are located at different LDBMSs.
For example: an Update Proposal Maker intends to "destroy" a house in the FDBS. Data concerning the house is not only found in the cadastral database, but also in the sewer and the electricity database. Once the house is torn down not only the data concerning the house itself becomes obsolete, but also the connections to the sewer and the electricity system. Therefore these connections in the FDBS should be passivated as well.

2.5 Distribution of Changes to Other Agencies

Often data is purchased from an acknowledged data supplier. This data is used by various clients as basis for their own system, e.g. a road database may be used as basis for an accident monitoring system, i.e. the data supplied is "enriched" by the clients.
Changes to the data by the data supplier can be distributed to the clients by sending Update Proposals to the clients, which may accept them into their LDBMSs.
This situation becomes more difficult when a client discovers that some data is out-dated. In this case not only the data supplier has to be notified, but the other clients using this data must be informed of the changes as well.

3 User Roles

To address the issues of distributed update in an environment with localised authority for update, one must carefully analyse the roles of the users; we differentiate between [3]:

3.1 The Update Proposal Maker

The Update Proposal Maker needs to retrieve information from the system, e.g. roads (road system, implemented in GEO), canalisation (canalisation database, implemented in ARC) or cadastral information. He is allowed to query or

browse the federated database.
The Update Proposal Maker first queries for the information, which may be located in LDBMSs outside his agency. Then he can modify the retrieved datasets, e.g. adding a piece of road locally and sends his update request, i.e. the Update Proposal to the authorised agent, i.e. the Validator.

3.2 The Validator

The Validator needs to check the proposals, which arrive from the different agencies. He uses his professional knowledge and the agency's rules to accept or reject the proposed changes to his LDBMS.
If the proposal is accepted, it is translated to the corresponding data-manipulation-statements and submitted to the FDBS. If not, the Validator has to send the Update Proposal back to the Update Proposal Maker, together with an explanation of what needs to be changed.

3.3 The Database Administrator

The Database Administrator is responsible for integrating new LDBMSs into the general schema and for maintaining this federated schema.

4 Architecture

This section describes the setup of a federated database environment suited for the proposed workflow. Please note, that this setup is rather general, as several forms of federated or even shared databases can be used to implement this particular workflow as long as a suitable form of metadata is provided. The only restrictions are that each tuple that may be updated can be unmistakably identified in the whole system and that there have to be clear distinctions between the responsibilities of the Validators.

4.1 The Physical Setup

As there are many different systems, using different data-formats, different data-structures, different Data Manipulation Languages (DML) etc., a way to logically connect these systems is needed. There has to be a way not only to connect the different systems via a network, but also to establish a logical distinction between user sites and the LDBMSs accessed by the users through the FDBS. Figure 1 shows the physical setup of such a system.
The physical setup consists of several connected sites. These sites each consist of a server, which holds its part of the FDBS, i.e., a departmental database, and the user sites. These user sites are used to retrieve and to manipulate data from the FDBS.

4.2 The Logical Setup

In order to distinguish between the FDBS on the one hand and the User Sites on the other, a logical separation of these entities is desired. The logical setup is given in Figure 2.

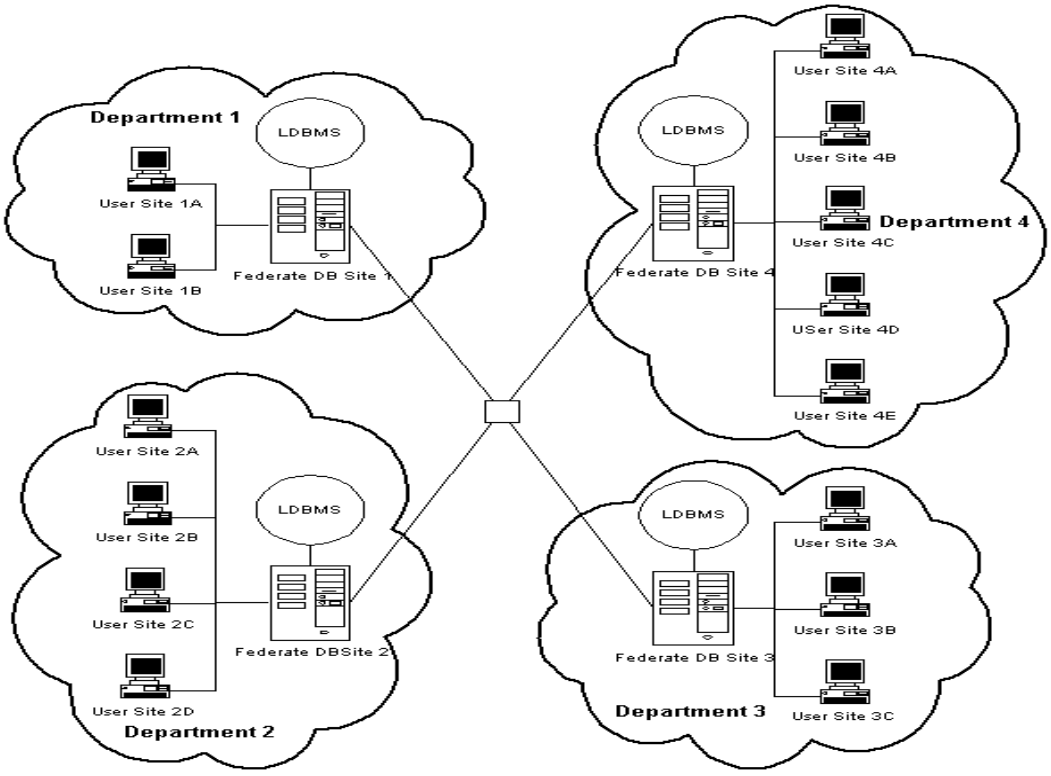

Fig. 1 The Physical Setup

4.2.1 *User sites*

The users of the FDBS-system are using these sites to retrieve needed data from the FDBS. User sites can be also used to prepare Update Proposals.

4.2.2 *Federated Database Sites*

The LDBMSs are located on the federated database sites. They appear as one large federated database to the clients. Physically the federated database sites act as servers to the client sites, with which they form a system that could work independently as well. These systems may consist of different vendors' products; e.g., Oracle, Adabas and Interbase databases are joining in one federated database.

When the FDBS is set up or whenever a new database is integrated into the FDBS, schema integration has to be performed. The task of this schema integration is to identify shared objects within the local databases and to provide a form of mapping between objects located in different local databases. This can be achieved by the use of metadata, which describes the shared objects in an abstract way and thus enables the system to identify objects within the whole system. More detailed discussions of schema integration can be found in [2, 10, 9, 4].

Another important aspect of the FDBS is the limited heterogeneity. This approach relies on identifiers for each entity. It is necessary that each tuple that may be updated can be unmistakably identified in the whole system. Another restriction is that all attributes affected by an update must exist in the proper format as well, e.g. it must not be possible to write characters into an integer attribute.

The federated database sites are also used to store the Update Proposals, which are submitted to the system as storable transaction scripts. After an Update Proposal is issued to the multi-agency system, it is stored at the concerned federated database sites. The Update Proposals are kept in a transitory state until the whole Update Proposal is either accepted or rejected by the Validators.

5 Workflow

The workflow within the multi-agency system is illustrated in Figure 3.

The Proposal Maker has to query the FDBS for the desired data. If he is permitted to extract the desired information, he receives the referenced datasets. This data may be extracted from LDBMSs using different query language facilities; therefore it may be represented in different data formats. So either the FDBS provides services that allow a unified access to this data or it has to be imported into the local system using appropriate tools.

The next step for the Proposal Maker is to build an Update Proposal using his local tools. By doing so he locally up-

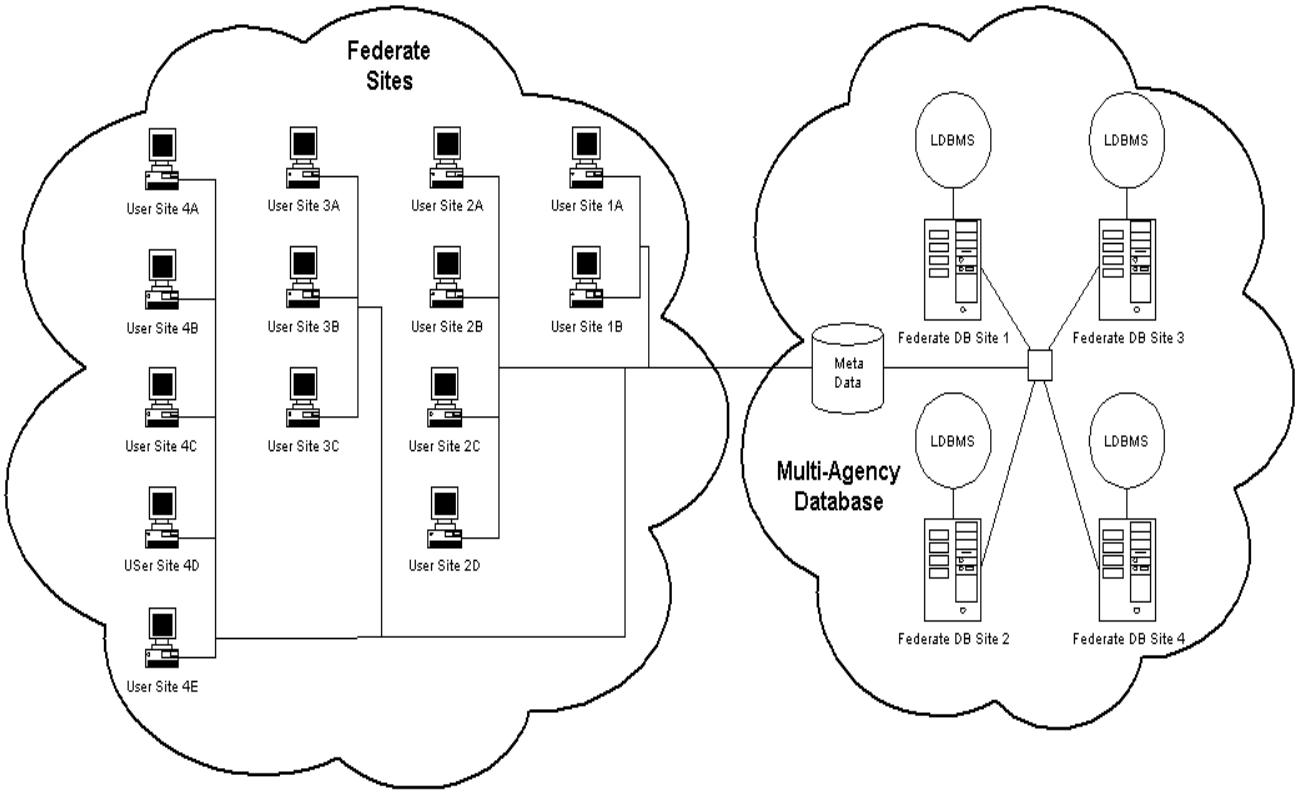

Fig. 2 The Logical Setup

Fig. 3 TheWorkflow

dates the extracted data. The changes applied to the local database are then represented in the vendor-independent Transformation and Transaction Management Language (TTML), i.e., a TTML script is generated. This generation uses mapping information derived from a meta-database to map the names of the updated objects to system-independent logical names. By doing so independence from local schema definition is achieved. The Update Proposal in TTML is then stored in the system together with additional information about which databases have to be changed, who initiated the change, by whom it has to be authorised, etc. Finally, the Update Proposal is sent to the sub-systems, i.e. the corresponding federated database sites holding the concerned LDBMSs. There it is stored in a transitory state, awaiting further processing by the Validators.

Validators can query which Update Proposals are awaiting validation and can deal with them individually, checking them against the agency's own documentation, which may be stored in the Validator's LDBMS, but is not available to the FDBS or other documents. The Validator may accept the Update Proposal after thorough examination of the proposed changes. The Validator may browse the FDBS if this is necessary. It is also possible that an Update Proposal may be checked by several Validators working at different agencies, i.e. different federated database sites. This can be necessary when an Update Proposal is about to change data that is used by several agencies and therefore stored in different LDBMSs.

After acceptance the Update Proposal is translated into the FDBS's Data Manipulation Language (DML). Then the resulting global transaction is executed by the FDBS. At this stage the Update Proposal may be accepted by the legal authorities, but it may still be rejected by the federated database, because the Update Proposal may violate constraints in one or more parts of the FDBS. It is the responsibility of the FDBS to guarantee that a global transaction's sub-transactions are committed and, if only one does not commit, to roll back the whole global transaction and to notify the Validator. In case of an error a description as detailed as possible should be provided to both the Validator and the Update Proposal Maker. If all sub-transactions commit successfully, the Update Proposal is permanently stored in a history database in order to keep track of the applied changes to the FDBS.

If the Validator decides not to accept the Update Proposal after thoroughly examining the proposed updates, because he cannot permit them, it is sent back to the Update Proposal Maker with an explanation. The Update Proposal Maker may also receive detailed information why the Proposal has not been accepted and thus be able to correct his Update Proposal accordingly and re-send it.

6 Application of the Workflow

This section describes the application of the proposed workflow to the motivating examples given in section 2.

6.1 Changes by Outside Agents

If an Update Proposal is submitted to one LDBMS, i.e. to one Validator, by an outside agent the Validator has to validate these changes and to eventually authorise them.

In case of the example given in section 2.1 this means that the original data in the road database is directly manipulated with data from another database and no digitising is necessary.

6.2 Conditional Validation

The example given in section 2.2 can be described with a list of pre-conditions: only if the cadastre database is changed appropriately the new house may be inserted in the FDBS. If a Validator encounters a situation like this he can issue a Conditional Validation. This special form of authorisation tells the FDBS that the Update Proposal may only be applied if another Update Proposal was successfully applied to the FDBS, i.e. validated and authorised by a Validator and successfully applied to the FDBS. Therefore the Update Proposal keeps pending until all pre-conditions are fulfilled.

6.3 Correction of Out-Dated or Incorrect Data

If an Update Proposal Maker or another user of the system encounters out-dated or incorrect data, he submits an Update Proposal to the concerned Validator.

As these faulty database entries are often detected while preparing an Update Proposal, the Update Proposal will most likely submit two Update Proposals: one that contains his original changes and one to correct the FDBS. The latter is most likely a pre-condition for the first Update Proposal and therefore a Conditional Validation for the first Update Proposal is issued.

6.4 One Proposal Changes Several Databases

If an Update Proposal would influence several LDBMSs it is divided into several Update Proposals, each containing all the information for one LDBMS. The concerned Validators validate these Update Proposals. If only one Update Proposal is not applied to the FDBS, i.e. the Validators' LDBMSs, either because it is not authorised by the concerned Validator or because it violates constraints

in the LDBMSs all sub-transactions of this global transaction have to be rolled back.

6.5 Distribution of Changes to Other Agencies

If a data supplier wishes to submit changes to its clients it sends corresponding Update Proposal to all concerned LDBMSs. If a client wants to change data in the global data stock it submits an Update Proposal to the data supplier without changing its own LDBMS. If the data supplier authorises these changes they are sent to all concerned clients.
In both scenarios problems may arise if due to the changes to the supplied data enriched data becomes invalid. In the given example of the accident monitoring system a road bend may have some associated attributes describing accidents at this specific point. If the bend is replaced by a piece of straight road in the global data stock these attributes become invalid. In this case manual correction of the entries in the LDBMS may become necessary. Using the proposed workflow it is at least possible to supply the Validator with information on which entries become invalid. Nevertheless, the correction of the associated enriched data cannot be fully automated.

7 Long-Running Transactions

As we have demonstrated Update Proposals are storable transaction scripts, which may remain in the multi-agency system for a longer period of time, sometimes even years. For instance, it may take months for the Validators to come to a decision whether to accept or to reject an Update Proposal or a decision depends on court action, which can take years to come to a final decision. During this time the update transaction is already stored in the system and awaiting further processing by the Validators. It is possible to query the database for the existence of a yet unauthorised Update Proposal. A Validator can also check Update Proposals by studying their effects using a viewing tool. Update proposals also fulfil the ACID-properties for transactions:
* Atomicity: An Update Proposal is either completely executed or not at all, i.e. either all sub-transactions of the LDBMSs are completed successfully or not by the FDBS.
* Consistency: As long as the underlying FDBS fulfils this criterion, so do Update Proposals.
* Isolation: The results of an Update Proposal are not visible to other transactions until the whole Update Proposal, i.e. all its sub-transactions, commits successfully. This is only possible if the underlying FDBS guarantees isolation.
* Durability: In this approach durability can be guaranteed if it is guaranteed by the underlying FDBS.
As long as the FDBS guarantees the ACID-properties, Update Proposals can be treated as transactions. They can even be characterised as a generalised form of long transactions, i.e. a transaction that needs a longer period of time to execute. Update proposals remain in the system for a longer period, but their actual execution time is as long as any global transaction in the FDBS. These Update Proposals can be viewed by the authorized personnel using appropriate viewing tools.

8 TTML

The intention of this section is to describe the TTML language, which provides a way to achieve system independence of updates to the federated database.

8.1 TTML Layer

Because TTML is used to represent updates of data in the FDBS, which consists of several different LDBMSs, an additional layer of abstraction is introduced.
The man-machine interface at the client site builds updates and then automatically translates them to TTML using a TTML generator. This means that the TTML generator has to map the functions of the local client to an abstract description of these functions. This is achieved by mapping the basic database functions to a system-independent abstract equivalent. The basic database functions used are: create, passivate and replace objects, create and passivate associations and replace attributes.
After generation the resulting TTML scripts are sent to the corresponding federate sites. After receiving a TTML script the objects referenced in the TTML script have to be mapped to the corresponding object names in the LDBMSs. In order to do this metadata describing these objects is used. The translated TTML script is then passed to the FDBS, which executes the resulting transaction by splitting it into several sub-transactions. These sub-transactions are then executed in the corresponding LDBMSs.
TTML uses objects, associations and attributes as the smallest units of representation. Using this form of representation TTML is capable of transporting data without any special adaptations to the data's semantic background. For example, TTML is able to represent geometric data as well as business data. The crucial point here is that the data schema has to be thoroughly analysed in order to identify these objects, associations and attributes semantically correct. The result of this analysis is then used to set up the Meta-Database, which is used by TTML generator, the TTML interpreters as well as by the FDBS.

8.2 Layout of TTML-Scripts

Each TTML-script consists of three parts:
* A TTML-header describing classes of objects, attributes and associations connecting two objects.
* TTML-statements describing the actions that are to be performed upon these objects, attributes and associations.
* TTML attached file: this part contains the data needed to

perform the TTML-statements.

8.3 TTML Header

The TTML header describes the data schema of the TTML exchange. It defines the referenced objects, attributes and associations, and relates them to unique identifiers within the whole TTML script. Note that the names used to identify the updated objects are only synonyms, because the same object may have different names and structures in different LDBMSs. The mapping logical to real names is done through a set of metadata describing the names and structures of all accessible objects within the FDBS. For example:

entity : class; ent_id 1; class_name Building

This statement describes the class "Building" and relates it to the entity identifier "1". In the same way associations are being described. Additionally to their attribute identifiers, attributes need entity identifiers in order to specify the class or association they belong to.

This example and the examples in the following sections are only schematic and do not comply with the correct syntax of TTML. They are given to demonstrate the general idea of TTML. Refer to [7] for the correct syntax.

8.4 TTML Statements

TTML statements describe which modifications are applied to the objects, attributes and associations described by the TTML header.

Objects can be created, passivated and replaced; values of attributes can be replaced, and associations can be created or passivated. Here too, object identifiers of newly created objects are only substitutes for the identifiers created during entering the transaction in the FDBS. For example:

Create_Object(Building DB,Building,01/01/98,10)

This statement creates a new instance of the class "Building" in the local database "Building DB", sets its object identifier to "10" and its creation date to "01/01/98". Note that the object identifier may differ from the class identifier. An object identifier is used to identify an object, whereas a class identifier is used to identify a class, from which objects may be instantiated. The same goes for associations and attributes.

8.5 TTML Attached File

This file contains the data needed to perform the TTML statements. It contains the values of the attributes, with which the referenced objects have to be populated. The distinction of actions (in the TTML statement script) and data (in the TTML attached file) is used, because data can also be imported using different data files, i.e. files describing values of objects, associations and attributes, using a different format. An example of an attached file statement:

Building :: 10 : OWNER : RUDI

Within the context of the examples in the previous sections, this statement describes that upon creation of the object of the class "Building" with object identifier "10", the attribute "OWNER" will be set to "RUDI".

9 Metadata

TTML is an abstract way to represent Update Proposals. By translating these Update Proposals to a form compatible with the FDBS, the proposed changes can be applied. But TTML needs information on the structure of the classes, associations and attributes referenced in a TTML script. Furthermore, there has to be a mapping from the symbolic names of the objects in the TTML statements to the names and locations of these objects in the FDBS. Note that this is not a trivial task, because an object in LDBMS 1 may refer to several objects in LDBMS 2. Setting up metadata is a matter of the integration of the pre-existing stand-alone LDBMSs into the FDBS. Therefore a very detailed analysis of the schema and thorough testing has to be performed in order to ensure that the mapping is correct and consistent. However, the integration only has to be performed for data that is shared in the FDBS. Data that is "private" for the agency must not be integrated into the FDBS.

10 Summary

We have proposed a solution to improve interoperability of databases used in public administration by addressing the workflow in a multi-agency situation, where the agencies share data in a Federated Database, but authority to permit updates is with each agency. In such situations it is necessary that another than the authorised agency can propose updates - but that it cannot commit them, lacking update permits for the database. The solution is to construct storable Update Proposals, which can be passed to the responsible Validator, who may authorise and execute them as updates to his agency's local database.

The method is based on the separation of the roles of Update Proposal Maker and Validator. Update Proposals are transmitted in the vendor-independent TTML language. Some time after reception the Validator checks if the proposed changes to the environment are to be permitted. If he permits the changes, the Update Proposal is translated into the appropriate data manipulation statements and sent to the underlying Federated Database, where it is executed as a global transaction.

Update proposals can also be seen as a generalised form of long-running transactions. The information transported by Update Proposals is available to authorised personnel as soon as the Update Proposals are submitted by the Update Proposal Makers. The integration of this information into the FDBS only takes place after the acceptance of the Update Proposal by the Validators, which may take years.

The translation uses the Commuter Metadata to translate the symbolic names used in the TTML-script into database object names. If only one update of the sub-transactions fails, because it would violate constraints implemented in the corresponding local database, all sub-transactions of this transaction to the FDBS have to be rolled back.

Acknowledgements

Support from the European Commission (DG III) for the Commuter Project and GIPSIE Project is gratefully acknowledged. We wish to thank our colleagues in the Commuter Project, especially Medur Sridharan, Francesco Fusco and Djahanguir Djamei.

References

1. Barghouti, N.S. and Kaiser, G.E.: Concurrency Control in Advanced Database Applications, ACM Computing Surveys, 23(3) (1991)
2. Castellanos, M.: A Methodology for Semantically Enriching Interoperable Databases, Advances in Databases - Proc. of the 11th British National Conference on Databases (1993)
3. Djamei, D. et al.: Architecture of the Commuter System, Commuter Consortium (1997)
4. Ekenberg, L. and Johannesson, P.: A Formal Basis for Dynamic Schema Integration, Conceptual Modeling - ER'96, Proc. of the 15th International Conference on Conceptual Modeling (1996)
5. Hwang, S.-Y., Huang, J. and Srivastava, J.: Concurrency Control in Federated Databases: A Dynamic Approach, CIKM'93, Proc. of the Second International Conference on Information and Knowledge Management (1993)
6. Korth, H.F. and Speegle, G.: Formal Aspects of Concurrency Control in Long-Duration Transaction Systems Using the NT/PV Model, ACM Transactions on Database Systems, 19(3) (1994)
7. Rayna, L. et al.: TTML Specifications, Commuter Consortium (1997)
8. Sheth, A.P. and Larson, J.A.: Federated Database Systems for Managing Distributed, Heterogeneous and Autonomous Databases. ACM Computing Surveys, 22(3) (1990)
9. Schmitt, I. and Saake, G.: Integration of Inheritance Trees as Part of View Generation for Database Federations, Conceptual Modeling - ER'96, Proc. of the 15th International Conference on Conceptual Modeling (1996)
10. Vermeer, M.W.W. and Apers, P.M.G.: On the Applicability of Schema Integration Techniques to Database Interoperation, Conceptual Modeling - ER'96, Proc. of the 15th International Conference on Conceptual Modeling (1996)
11. Chirie, F. and Frank, A.U.: Scenarios for Demonstrating the Commuter Federated Multi-Agency Capabilities, Commuter Consortium (1998)