Integration Ontology for Distributed Database

Ana Muñoz[1], Jose Aguilar[2], and Rodrigo Martinez[3]
1 Instituto Universitario Tecnológico de Ejido. Mérida Venezuela
anamunoz@ula.ve,
2 Uiversidad de Los Andes. CEMISID. Mérida Venezeula.
aguilar@ula.ve
3 Uiversidad de Murcia. Murcia España.
rodrigo@um.es

Abstract. In this work we will study the problem of the design of the "Integration Model for Distributed Database System". We particularly design the canonical model through the ontological handling of the information. The ontology is designed in a way that allows the description of a database like a set of representative terms of its different components. In this ontology, the definitions use classes, relations, functions, among other things, of databases, to describe their components, operations and restrictions, as well as, the process of integration. These databases can be Relational, Fuzzy, Intelligent and Multimedia.

1 Introduction

The interoperability between different systems information is one of the most critical aspects in the daily operation of many organizations. In the last decade this preoccupation was increased with the proliferation of different databases, with different data models, that run in different platforms. The systems of distributed databases, also known as federated databases, allow to have available the information from different sources of intelligence that can be heterogeneous, distributed and independent. A federated database acts like a front-end application of manifold component. The federated database provides operations for the access to each component, maintaining the consistency of information between the diverse sources and providing a uniform access method to the services that each component offers.
The diversity of programming languages, data models and methods of integration, determine different styles in the architecture for a federated database, that varies from a loosely coupled to tightly coupled approach. In general, the tightly coupled systems integrate the diverse sources of intelligence through a global conceptual scheme, normally denominated canonical model, providing a uniform vision of the diverse components at a high level. The use of a canonical model hides the structural differences between the different components and gives to the user the illusion to be accessing a simple centralized database. On the other hand, on the

systems tightly coupled the integration of the components is based on a language of common access that all the components must decide, in a way that all the functions are standardized.
In this work we will deal with the design of the "Canonical Model for Integration of Distributed Databases". Particularly, we set out to design the canonical model through the ontological handling of the information. This ontology allows describing a database like a set of terms that represent its different components. In this ontology, the definitions use classes, relations, functions, among other things, of the databases, to describe its components, restrictions, operations, etc. The reason of using ontologies is that they define concepts and relations within a taxonomic frame, whose conceptualization is represented, of a formal way, legible and usable. Of this form, ontology is a common and shared understanding of a domain that can be used to communicate heterogeneous systems [7].
The integration of tightly coupled federated database has been treated in previous works for relational and objects databases. Alvarez in its work presents a proposal of binary integration for the generation of a federation of component databases [1]. In addition, it presents a scheme to use the local components through a query language. In the work of Abello et al., they present an integration model in real time to databases using the canonical model BLOOM [2]. These works use the architecture for federated databases of Shet&Larson [10]. In previous works [8] we have represented an architecture for the integration of database where it is necessary a canonical model.
Like continuation of that work, in this article the ontological taxonomies that compose the databases integration architecture are described, and the Canonical Model is designed using this ontological notion. This way, the processes of integration of the different types of databases and of resolution of conflicts are defined through the ontology. In addition, the integration ontology is translated to first-order logic predicate, so that from it we design the mechanisms of consultation, update and data mining for Intelligent Distributed Database. This article shows in one first part, the theoretical aspect on which the same one is based, which includes to the distributed databases, as well as the ontology concepts. In the second part the integration process is described through ontological schemes, as well as its axioms that defines the logic expressions of the integration process. The ontological schemes of the component databases are described in other work [11]. Thus, the fundamental aspect of this work is to propose a ontological frame based on sentences of First-Order Logical Predicate (LPO) for the integration of a federation of databases.

2 Theoretical Aspects

2.1 Distributed Databases

The distributed databases talk about the integration of necessities of no local storage and processing where is necessary to interchange originating information of different sites [1, 2]. The systems of distributed databases integrate systems of diverse databases, to give to the users a global vision of the information available. The decentralization of the information promotes the heterogeneity in its handling. This can occurs in many levels, from the form and meaning of each data to the format and the storage media that are chosen to keep it. From the functional organization, the systems of distributed database are divided in two classes: A homogenous distributed database that is a collection of multiple data. The homogenous systems are looked like a centralized system, but instead of storing to all the data in a single place the data are distributed in several sites communicated by the network. The heterogeneous systems are characterized to handle different database in each node. An important subclass is the Federated Databases, which integrate information from heterogeneous databases, and present a global access to the users, with transparent methods to use the total information in the system. The main characteristic is the autonomy that the local

databases, also called Component Databases, conserve. In order to build the federation of Component Databases, it is necessary to provide a mechanism that is able to obtain a global scheme of databases, which allows a transparent access to the different databases existing [10]. The heterogeneity in the component databases can be presented in several aspects: hardware, software, data modeling, and semantic aspect, among others. A System of Federated Database (SFDB) is classified like weakly connected or strongly connected, based on the idea of whom handles the federation and how their components are integrated. A SFDB is weakly connected if the responsibility to create and to maintain the federation falls to the user, and there is not control on the part of the federated system and its administrations. A federation is strongly connected when the federation and its administrators are responsible for the creation and the maintenance of the same one, and participate actively in the control of the Component Database. A strongly federated system connection can be of two types: With unique federation, if it allows the creation and management of an only federated scheme. With multiple federations, if it allows the creation and management of multiple federated schemes. Each SFDB has an architecture of schemes to surpass the syntactic and semantic heterogeneities. Shet&Larson [10] proposes an architecture of schemes for a SFDB composed by: i) *Local Scheme*. It is the conceptual scheme of the Systems of Component Database that integrates the Federation; ii) *Component Scheme*. The conceptual schemes of the component databases are translate to a canonical model, that is a common data modeling for all the databases that are going to compose the federation; iii) *Scheme of Export*. In this scheme is described the part of the component schemes that are going to be shared as well as their location and access control; iv*) Federated Scheme*, in this scheme is made the integration of the multiple schemes of exportation; v) *External Scheme*. This is the scheme for each user and/or application of the SFDB.

2.2 Canonical Model

The ability of representation of the database comes given by its data modeling. A data modeling is made up of structures, operations and the restrictions in the use of them. The ability of representation of a data modeling is made up of two factors[9]: i) *Expressivity***.** The expressivity of a data modeling is the degrees in which a model can directly represent the concepts that it conform. ii) *Semantic Relativism*. The semantic relativism of data modeling is the power of its operations to derive external schemes.

When different databases form a federation, they require a integration data modeling, called Canonical Data Model (CDM). The CDM is the element that processes the query and updates that are made to the federation. Thus, following the architecture of five levels of Shet&Larson [10], we can develop a common CDM to all the federation. The use of a CDM solves the problem of syntactic heterogeneity, consequence of the use of different native data models. The heterogeneity semantic, resulting of different conceptualizations from Component databases, is solved in the process of integration of schemes. The CDM has the following characteristics: i) Generalization: it is the process by means of which, from two or more entities is constructed a new entity; ii) Association: it defines a new entity from the relations between two or more entities; iii) Classification: allows to group entities in classes, that is constructs a new entity from the common characteristics of other entities. The CDM must support the definition of new operations and restrictions, must allow the implementation of integration operators, among other things [9]. We will use ontologies to represent our CDM, since they allow integrating databases using intelligence during the process of conformation of the federation, as well as the semantic enrichment through the integration of the databases with its concepts, operations and restrictions.

2.3 Ontology

A definition of Ontology in terms of database is the following [4, 7]: "***Ontology is a database that describes the concepts of the world of some domain, some of its properties and how these concepts are related between them*** ". The knowledge represented within ontology is

formalized through five components: i) *Concepts or classes*: They are the ideas to be formalized. They belong to a certain domain of application, and can be organized in taxonomies; ii) *Relations*: They represent the interactions between the classes and are defined as a subgroup of a Cartesian product; iii) *Functions*: They are a special case of relations, where elements are generated by means of the calculation of a function; iv) *Instance*: they are used to represent elements or individuals in an ontology; v) *Axioms*: They serve to model sentences that always are going to be certain. They are used to represent knowledge and are used to represent the properties that concepts and instances must satisfy. For example: If animal class animal is mammalian; the instance dog is mammalian.

Classifications of ontologies have been done in agreement with the type of concept to describe and its use [4, 5]: i) *Terminological*: they specify the terms that are used to represent knowledge. Usually they are used to unify vocabulary in a certain domain; ii) *Knowledge Modeling*: they specify concepts related to the knowledge. They contain a rich internal structure and usually are fixed to the particular use of the knowledge that they describe; iii) *Ontologies of domain*: These ontologies are specific for a domain in concrete; iv) *Ontologies of tasks*: These ontologies represent the tasks that are susceptible to make in a domain in concrete; v) *General Ontologies*: They represent general information and nonspecific of a domain.

3 Design of an Intelligent Model of Integration for Federated Databases

The design of our CDM will be based an Ontologies. These ontologies describe to each one of the databases to integrate, as well as the integration process. In the following figure is shown our Intelligent Canonical Model (modeled in a Knowledge and Facts Database), and that has learning and reasoning mechanisms to carry out the integration process.

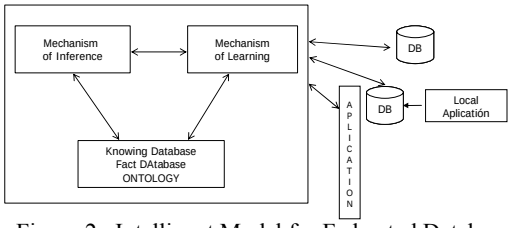

Figure 2. Intelligent Model for Federated Databases

The Federated Databases integrate information from local heterogeneous databases and allow the global access to the users. The main characteristic is the autonomy that the local databases or Component Databases conserve. In order to allow on a federation of Component Database, we need to provide an integration mechanism for obtaining a global approach of the resources of information of an organization. This is obtained through the canonical model.

3.1 Concepts of Federated Databases

A Federated Database is a component database that has operations and restrictions of integration. The Component Databases are the databases that conform the federation. In our case, these component databases can be: Object-oriented Databases, Relational Databases, Multimedia Databases, Fuzzy Databases, or Intelligent Databases; also a component database can be another federated Database. Each one of these component databases has their concepts, operations and restrictions. In figure 3 is shown the ontological scheme that describes the concepts of the federated databases.

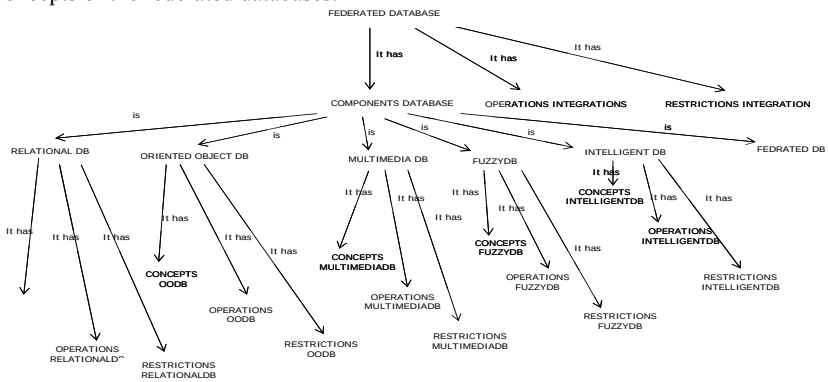

Figure 3. Ontological scheme of the components that integrate an Intelligent Distributed Database

In table 1 is described the ontological scheme of the figure 3 through axioms. These are used to define the ontology like logic expressions. Each axiom includes its description in natural language, and its logical expression.

Table 1. Axioms for the concepts of the Federated Databases

Sentence	LPO
A Federated database has component databases, and operations and restrictions of integration	∀ x FederatedDB(x) => Has (x,ComponentDB) Λ Has (x,IntegrationOperation) Λ Has (x,IntegrationRestriction)
The component databases can be relational databases, object-oriented databases, multimedia databases, fuzzy databases, intelligent databases and federated data bases	∀ x ComponentDB(x) => Is(x,RelationalDB) V Is (x, OODB) V Is (x, MultimediaDB) V Is (x, FuzzyDB) V Is (x, IntelligentDB) V Is (x, FedratedDB)
The Relational database has Concepts, Operations, and Restrictions	∀ x RelationalDB(x) => Has(x,ConceptsR) Λ Has(x, OperationsR) Λ Has(x, RestrictionsR)
The OODB has Concepts, Operations, and Restrictions	∀ x OODB(x) => Has(x,ConceptsOO) Λ Has(x, OperationsOO) Λ Has(x, RestrictionsOO)
The Multimedia Database has Concepts, Operations, and Restrictions	∀ x MultimediaDB(x) => Has(x,ConceptsMM) Λ Has(x, OperatinsMM) Λ Has(x, RestrictionsMM)
The fuzzy database has Concepts, Operations, and Restrictions	∀ x FuzzyDB(x) => Has(x,ConceptsFuzzy) Λ Has(x, OperationsFuzzy) Λ Has(x, RestrictionsFuzzy)
The Intelligent Database Concepts, Operations, and Restrictions	∀ x IntelligentsDB(x) => Has(x,ConceptsInt) Λ Has(x, OperationsInt) Λ Has(x, RestrictionsInt)

3.2 Operations of Integration in a Database Federation

We will use the operations of integration according to Batini and Lenzerini [3, 6], which is made in phases. Next the characteristics of these phases are described.

Preintegration. In this phase is defined the order of integration of the databases and the parts of the databases to integrate. The integration order can be binary when two schemes are integrated simultaneously, and n-Aryan when they integrate n schemes simultaneously. Also, the policies of integration as far as the access restrictions and priority in the access to Component databases are defined. This procedure is the same when we form a new federation or when we can incorporate a component database to an existing Database Federation.

Comparison of the schemes. The databases are compared and analyzed to determine the correspondence between concepts and to detect the possible conflicts. Once the conflicts are

detected, they are sent to the Conflicts Management System to solve them through a system of rules.

Union and Reconstruction. Once solved the conflicts, the union of the different schemes from the component databases is made. The goal of this activity is to conform or to align schemes to make them compatible for its integration. It has operations like: transform an atomic concept into another one, eliminate redundant relations, create hierarchy of generalization.

In figure 4 is shown the ontological scheme that describes the operations of integration of a Database Federation.

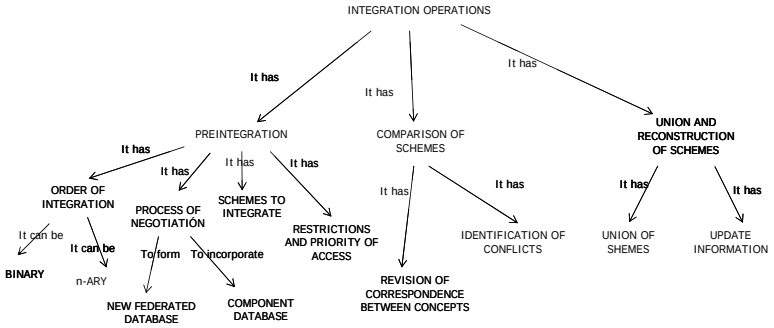

Figure 4. Ontological Scheme to Operations of Integration for a Database Federation

The Axioms for the operations of integration of a Database Federation are in the table 2:

Table 2. Axioms for the operations of a Database Federation

Sentence	LPO
The operation of integration has the phase of preintegration, comparison of schemes and conformation of the canonical model	∀ x OperatiónIntegration(x) => Has (x,Preintegration) Λ Has(x,ComparisonSchemes) Λ Has (x,ConformationCM)
The Preintegration defines the integration order, the negotiation process, the schemes to integrate, the restrictions and the priority of access	∀ Preintegration(x) =>Has(x, OrderIntegration) Λ Has(x,ProcessNegotiation) Λ Has(x,SchemestoIntegrate) Λ Has(x, RestrictionsofAccess) Λ Has(x, PriorityofAccess)
The Order of integration of the databases can be binary or n-Aryan	∀ x OrderIntegration(x) => Is(x,BinaryIntegration) V Is(x, n-aryanIntegartion)
In the Process of negotiation a new federation is formed or a component database is added to an existing Database Federation	∀ x ProcessNegotiation(x) => Formed(x,NewFederatedDB) V Added(x, ComponentDB)
In the comparison of schemes must be reviewed the correspondence between concepts to determine the conflicts	∀ x SchemesComparison(x) => Has (x,ReviewCorrespondencebetweenConcepts) Λ Has(x,IdentificationofConflictsIntegartions)
A binary order of integration integrates two schemes simultaneously	∀ x BinaryIntegration(x) => Integrate(x,TwoSchemes)
The integration order n-Aryan is the one that Integrate n schemes simultaneously	V x N-AryanIntegration(x) => Integrate(x,NSchemes)
The access restrictions are the authorizations to accede to the component databases that conformed the federation	∀ x RestrictionsAccess(x) => ItAuthorizes (x,AccessComponentDB)
The access priority establishes the order of access to the component databases	∀ x AccessPriority(x) => Establishes(x,OrderofAccesstoComponentDB)

The union and reconstruction of schemes define the union of schemes and the update of the information in the model	∀ x UnionandReconstructionSchemes(x) => Have(x,UnionSchemes) Λ Have(x,UpdateofInformation)

3.3 Restrictions of Integration in a Database Federation

In the integration of the databases, the following types of conflicts can appear:

- Conflicts in Tables: Conflicts in the Name of tables, Conflicts in the Structure of the tables, objects and multimedia elements, Conflicts in the Restrictions of Integrity.
- Conflicts of Attributes: Conflicts in name of Attributes, Conflicts in Values by Default, Conflicts by Restrictions of the Attributes Values, Conflicts by the Cardinality and degree of Atomicity, Conflicts in the Representation of the Information.
- Conflicts of Data: Conflicts between the values, when equivalent instances have different values because the collected data are incorrect or are obsolete. Differences in the representation.
- Conflicts in Rules: Simultaneous firing of Rules, Contradiction between rules.

In figure 5 is shown the ontological scheme that describes the conflicts.

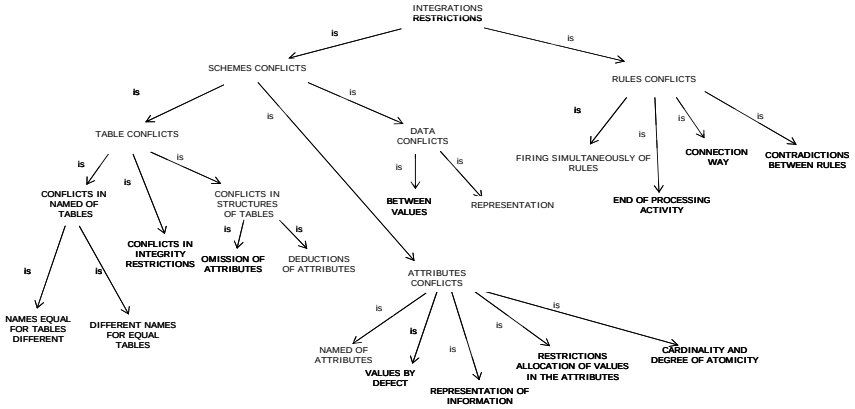

Figure 5. Ontological scheme of the Integration Restrictions for a Database Federation.

The Axioms of the restrictions of integration for a Database Federation are in table 3:

Table 3. Axioms for the restrictions in a Database Federation

Sentence	LPO
The integration restrictions can be conflicts in schemes or conflicts in rules	∀ x Integrationrestrictions (x) => Is(x,ConflictsSchemes) V Is(x,ConflictsRules)
The conflicts in scheme can be conflicts in tables or conflicts in attributes or conflicts in data	∀ x ConflictsSchemes(x) => Is(x,ConflictsTable) V Is(x,ConflictsAttributes) V Is (x,ConflictsData)
The conflicts in tables can be in name of tables, structure of table, of object or of multimedia element or in integrity restrictions	∀ x ConflictsTable(x) => Is (x,ConflictsNamedTable) V Is (x, ConflictsStructureTable) V Is (x,ConflictsStructureObject) V Is(x, ConflictStructureMM) V Is (x,ConflictRestrictionIntegrity)
The tables name conflicts arises when different names for equal tables or equal names for different tables exist	∀ x ConflictsNamedTable (x) => DifferentNamedTables(x,EqualTables) V EqualNamedTables(x,DiferentsTables)
The conflict in table structure happens when there are attributes that are omitted or when there are attributes that are deduced	∀ x ConflictsStructureTable (x) => Is(x, AttributesOmitted) V Is(x,AttributesDeduced)

The conflict in structure of Object happens when there are attributes of the object that are omitted or when there are attributes of the object that are deduced	∀ x ConflictsStructureObject(x) => Is(x,AttributesObOmitted) V Is(x,AtributtesObDeduced)
The conflict in multimedia structure happens when there are attributes MM omitted or when there are attributes MM that are deduced	∀ x ConflictStructureMM => Is(x,AttributesMMOmited) V Is(x,AttributesMMDeduced)
The conflicts in attributes can be conflicts in name attribute or conflicts in values by default or conflicts of restrictions of values of the attributes or conflicts of cardinality or conflicts in the representation of the information	∀ x ConflictsAttributes(x) => Is(x,ConflictoNombreAtributo) V Is (x,ConflictsValuesByDefault) V Is (x,ConflictsRestrictionofAlocationsofValues) V Is (x,ConflictsCardinality) V Is (x,ConflictosRepresentactóonInformation)
The conflicts in name of Attributes has different names for equivalent attributes or equal names for different attributes exist	∀ x ConflictsNomedAttributes(x) => HasDifferentNames(x,EquivalentAttributes) V HsEqualNames(x,DiferentAttributes)
The conflicts in values by default occur by definition of the values deduced by default	∀ x ConflictsValueByDefault(x) => Has(x,DefinitionOfValuesDeduced)
The conflicts by Restrictions of Values to the Attributes can be conflicts in the data types and conflicts in the domain of restrictions.	∀ x ConflictsRestrictionofAllocationofValues(x) => Is(x,ConflictsinDataType) V Is(x,ConflictsinRestrictionsofDomain)
A cardinality conflict is the difference of details of the attributes	∀ xCardinalityConflict(x) => Has(x, DifferentLevel fromRepresentationofAttributes)
The Conflicts in the representation of information are the different domain that an attribute represents	∀ x RepresentationofInformacionConflict(x) => Has(x,DifferentDomain)
The conflicts in data can be conflicts between values or conflicts of differences in the representation	∀ x ConflictsData(x) => Is (x, ConflictsbetweenValues) V Is (x, ConflictsofDifferencesintheRepresentation)
A conflict between values arises when equal instances have different values	∀ x ConflictsbetweenValues(x) => Has(x,EqualInstancesofData) Λ Has(x, DifferentValuesofData)
The representation differences has different representations for a same data	∀ x ConflictofDiferencesofRepresentation(x) => Has(x,DifferentRepresentationOfDifferentData)
A conflict in rule can be a firing simultaneously of rules, or can be a conflict in the connection way or can be conflict in the aim of the processing of rules, or can be a contradiction between rules	∀ x ConflictinRule(x) => Is (x, FiringSimultaneouslyofRules) V Is (x, ConflictintheConnectionWay) V Is (x,EndOfProcessing) V Is (x,ContradictionBetweenRules)
A simultaneous firing of rules is when an event activates more than one rule	∀ x SimultaneousFiringofRules (x) => Isa(x, ShootsmorethanoneRule)

4 Conclusions

In this work the ontological schemes that represent the process of integration of databases are presented, based on the architecture of Shet&Larson [10] for federated databases. The development of the ontologies is used like scheme that allows making the intelligent

integration of a federation of databases. Particularly, the canonical model must have the ability of representation of the different data models from level of its structures, operations and restrictions of the databases which conform the federation, solving the heterogeneity problems that can be presented. We use ontology like representation of the canonical model, since it allows taxonomically to describe the concepts in the domain of the databases and its properties. In addition, with the ontology we will be able to design management systems based on mechanisms of reasoning and learning. Thus, our Model of Intelligent Integration of Federated databases is intelligent and extensibility. In our representation of the Model of Intelligent Integration of Federated Databases we found the taxonomies that describe the concepts, operations and restrictions of the process of integration of the databases. The axioms interpret the taxonomy and will allow translating the ontologies to a language of knowledge. With them, new knowledge could be obtained and extracted.

In the future, a language of manipulation of the Intelligent Distributed Database will be designed using our ontology. For this, an inference mechanism must be designed that allow to reason during the processes of query and update over the Distributed Database. In addition, a mechanism of manipulation of the Canonical Model must be designed (learning) to update the knowledge. Also, from the inference mechanism tasks of data mining will be able to be done, such as generate patterns of access of users of the system to create virtual communities, extract new knowledge derived from the integration of the databases, etc.

References

1. Alvarez Carrión, G.; "Integración de esquemas en bases de datos heterogéneas fuertemente acopladas". Master thesis, Universidad de las Américas, Puebla. México 1999
2. Abello A., M. Oliva, J. Samos, and F. Saltor; "Information System Architecture for secure Data Warehousing". In Proc. of the 3rd Int. Workshop on Engineering Federated Information Systems (EFIS), pag. 33-40. 2000
3. Batini C., Lenzerini M.; "A comparative analysis of methodologies for database schema integration", ACM Computing Surveys 17, 4, December 1976.
4. Bertino E., Catania B., Zarri Gian P.; "Intelligent Database System", Addison-Wesley. 2001. http://ksi.cpsc.ucalgary.ca/KAW/KAW97/blazquez/
5. Corcho O., Fernandez-López M., Gomez-Perez A., "Methodologies, tools and languages for building ontologies. Where is their meeting point? Data & Knowledge Engineering 46 (2003) 41-64. Elsevier.
6. Fernandez-Breis J., Martinez-Béjar R.; "A cooperative framework for integrating ontologies"; Elsiever Science Human Computer Studies 2002.
7. Gruber, T. R. "A Translation Approach to Portable Ontology Specifications. KSL Report", 1993, http://ksl-web.stanford.edu/abstracts_by_author/Gruber,T..papers.html
8. Muñoz A., Aguilar J.; "Architecture for Distributed Intelligent Databases". IEEE, 13th Euromicro Conference on Parallel, Distributed and Network-based Processing, Euromicro-PDP 2005, pp 322-327
9. Saltor F., Castellanos M, García-Solaco M; "Suitability of data models as canonical models for federated databases"; Universitat Politècnica de Catalunya.
10. Shet P, Larson J., "Federated Database System for managing distributed, heterogeneous and autonomous databases". ACM Computing Surveys 22, 1990 pp 173, 236