A General Ontology for Intelligent Database

Muñoz Ana[,2], Aguilar Jose[2]

[1]Departamento de Tecnología, Área Informática

Instituto Universitario Tecnológico de Ejido

[2]Centro de Estudios en Microelectrónica y Sistemas Distribuidos (CEMISID)

Universidad de Los Andes

VENEZUELA

anamunoz@ula.ve, aguilar@ula.ve http://www.ula.ve/cemisid

Abstract— The intelligent Databases (IDB) are originated from the integration of databases technologies with artificial intelligence technologies. The IDB are characterized by the presence of stored rules in a rules base and facts stored in a facts base, all together conforms the knowledge base, in which different forms of reasoning are applied. In general, an ontology is a knowledge base that describes the concepts of a domain, their properties and their relations, providing a common vocabulary in a defined area. This article proposes an ontology for IDB that describes the concepts, operations and restrictions of these databases. Also, at the end of this paper we present an utilization example and its implementation using Protégé.

***Keywords*—** Ontology, Intelligent databases.

I. Introduction

The intelligent databases (IDB) have as general purpose the generated and the discovery of information and knowledge. Among these types of databases we include the active, deductive, knowledge and fuzzy databases. In general, the IDB are the natural evolution of the traditional databases, not only because they allow the manipulation of the data, also of the cognitive elements in form of facts and rules. One essential aspect of these databases are the possibilities of using techniques to discover knowledge, such as data mining techniques; all this permit learning patterns and data analysis strategies, as well as making classification and recognition, among others.

The IDB systems are characterized by using an artificial intelligent technique that supports different reasoning mechanisms, they have a similar architecture to the expert systems that consist of a fact base, a rule base and must have persistence of the fact base.

In this work, we design an ontology for an IDB that allows describing it as a set of representational terms of their different components. In this ontology, the definitions associate types, relations, functions, among others, in the universe of the speech of the IDB, in order to describe its meaning, its components, operation and restrictions. The reason of using ontologies is that they define concepts and relations within a taxonomic frame, whose conceptualization is represented in a formal, legible and usable way. This way, ontology allows a common and shared understanding of a domain [3, 5].

Manuscript received October 9, 2006. Revised version received May 1, 2007; Second Revised June 30,2007, work was supported in part by the U.S. Department of Commerce under Grant BS123456 (sponsor and financial support acknowledgment goes here). Paper titles should be written in uppercase and lowercase letters, not all uppercase. Avoid writing long formulas with subscripts in the title; short formulas that identify the elements are fine (e.g., "Nd–Fe–B"). Do not write "(Invited)" in the title. Full names of authors are preferred in the author field, but are not required. Put a space between authors' initials.

F. A. Author is with the National Institute of Standards and Technology, Boulder, CO 80305 USA (corresponding author to provide phone: 303-555-5555; fax: 303-555-5555; e-mail: author@ boulder.nist.gov).

S. B. Author, Jr., was with Rice University, Houston, TX 77005 USA. He is now with the Department of Physics, Colorado State University, Fort Collins, CO 80523 USA (e-mail: author@lamar. colostate.edu).

T. C. Author is with the Electrical Engineering Department, University of Colorado, Boulder, CO 80309 USA, on leave from the National Research Institute for Metals, Tsukuba, Japan (e-mail: author@nrim.go.jp).

II. Theoretical Background

A. Intelligent Databases

In [9] defines IDB as "*a database that contains knowledge about the content of their data. A set of validation criteria are stored with each data, for example maximum and minimum value or a list of the possible input*". Particularly, inside the concept of IDB the following technologies are included: knowledge based systems or experts systems, deductive database and active database, which are described in the next paragraphs.

1) *Knowledge based systems*

The Knowledge Based Systems (KBS) are applications than generate satisfactory solutions o answers to problems that require a reasoning by computer that involves knowledge of some type. Some type of Knowledge can be facts (that express valued proposals) or rules [2, 4]. The KBS construct its reasoning to solve problems concatenating affirmations and rules in line of reasoning. This reasoning lines show how a supposition set and specific set of assertions and rules produce a particular conclusion. Some of the KBS basic characteristics are the implicit representation of knowledge, the capacity of independent reasoning of the specific application, the capacity of explaining their conclusions and the reasoning process. The KBS base their yield on knowledge quantity and quality in a specific domain [2, 4]. The main elements of the knowledge based systems are: i) *Knowledge base (rules and facts):* It's a Knowledge representation of the system domain, ii) *Inference Machines:* It's a reasoning process from input data taking like the source of this process the knowledge base. iii) *Interface with the user:* inputs and outputs of the system, generally including answers and explanation mechanisms.

2) *Deductive Databases*

A deductive data base consists of two components:

- A dataset, called *facts*, representing specific information given by the user; these data are called collectively an extensional database (EDB).
- A set of inference rules, called *rules*, codified according to the domain knowledge, from which data can be derived using the facts; these rules are referred as intentional data base (InDB).

The different architectures for deductive databases are categorized according to the cooperation between the InDB and the EDB [2]: i) a *homogenous* architecture, in which are used a simple integrated system to manipulate the EDB and the InDB, and the deductive reasoning is made on them. ii) A *heterogeneous* architecture, in which are used relational database to manage an EDB, and a logical programming system is used to make a deductive reasoning.

3) *Active Databases*

An active database reacts automatically to events and supports the ECA rules (Event-Condition-Action). The occurrence of several types of events (transition, time events and external signals) shoots the evaluation of the conditions. If an evaluated condition is certain it carries out the action [1].

In general, each time it detects the occurrences of an event it notifies to the responsible component of the rule execution, this is called event signaling. Therefore, all the rules that are defined to respond to this event will be executed. The rule's execution implicates condition evaluation and action execution.

An active database has all the characteristics of a passive database (model, query language, multi-user access and recuperation characteristics). The use of ECA rules implies the following characteristics:

- **Event types.** A type event (description of event, pattern and definition) describes situations that have a reaction. An event type could be primitive or composed. A primitive event type defines elemental occurrences, for example: method's invocations, data modification, transactions, etc. The composed event type is defined as combinations of others events, primitive and composed, using a set of events constructors such as disjunction, conjunction, sequence, etc. The events occurrences are the instances of the event type.
- **Meaning of the conditions**. A condition formulates in which status the database must execute the action. An action formulates the reaction to an event and is executed after rules fire. An action could contain data modification, transaction operations, methods/ procedure call, etc.

B. *Ontologies*

A definition of ontology made in database terms, is the one that Weigand offers [3, 5] "*An ontology is a database that describes the world's concepts of specific domain, some of their properties and how these concepts relate among them*". The knowledge represented inside an ontology is formalized trough five components:

- *Concepts or classes*: they are the ideas to formalize. They are all the important ideas relevant to a certain domain of application and they can be organized in taxonomies. They can be descriptions of objects, tasks, functions, actions, strategies, groups, etc. For example, the animal concept.
- *Relations*: Represent the interaction between classes, and are defined as a Cartesian product subgroup. *Functions*: They are special relation cases, where it generates elements by mean of function calculation. For example: Price_Object= value+revenue+tax
- *Axioms*: They are used to model sentence that always are going to be certain. They are used to represent knowledge. They will be declaring theorems that must fulfill ontology elements. That is, they are defined theorems about the relations that all the elements of an ontology must have.

Van Heijst [10] proposes an ontology classification according to the concept to describe and their use:

- *Terminological*: specified terms used to represent speech universe knowledge. Usually they are used to unify vocabulary in a certain domain.
- *Information:* It offers a structure for the standards information storage.
- *Knowledge Modeled*: They specify related concepts to the knowledge. Contains a rich internal structure and usually they are fit to the particular use of the knowledge they describe.

III. Ontology for Intelligent Database

We will consider inside the IDB: the active database, the deductive database and the Knowledge based systems [6, 7, 8]. Figures 1 shows an ontological scheme for the IDB, from the taxonomic point of view, where concepts and relations are shown. The concepts are each of the node**,** the relations are the etiquettes on arrows. On the other hand, functions and axioms are represented through the first order predicate logic sentence. Those are shown on the tables. Now we present the concepts and relations of the proposed ontology for the IDB.

Concepts: INTELLIGENTDATABASE, IDBCONCEPTS, IDBOPERATIONS, IDBRESTRICTIONS

Relations: has

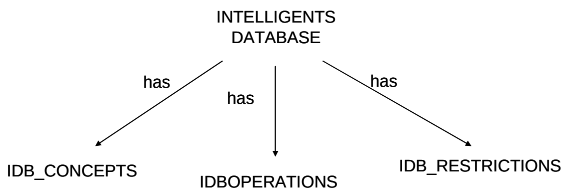

Figure 1. Ontological Scheme for IDB

The IDB have concepts that define the elements that conform it, operations that can be made and restrictions that define rules behavior. Table 1 shows the sentence that conform the ontological general scheme of the IDB. The IDB has the following attributes:

Intelligent Database (ID_BDI, Name _BDI, Address, Domain, Scheme, Model), where:

ID_BDI: IDB identificator, it is unique and allows identifying each database

Name_BDI: database name

Address: database electronic address that define the place where the intelligent database is located for possible actualizations and queries.

Domain: IDB domain, allows identifying in which data and knowledge areas they work.

IDB Scheme: description of the database, where it shows tables, datatype and relations among them, like it dictionary.

IDB Modeling: data model used by the IDB to describe schemes, such as relational, oriented object or semantic model, among others.

TABLE I
ONTOLOGICAL SCHEME OF THE IDB LIKE AXIOMS

SENTENCE	LPO
A IDB has concepts, operations and restrictions	V x IDB(x) => has(x, IDBConcepts) Λ has(x, IDBOperations) Λ has(x, IDBRestrictions)

The intelligent databases have concepts that define the elements that compose it, operations can perform and restrictions that define the behaviours of the rules and operations of intelligent databases.

A. Intelligent Database Concepts

In general, the IDB has two concepts: knowledge base and a reasoning mechanism. Thus, are knowledge based systems that by means of a reasoning scheme determining the rules that are activated until obtaining an answer to a certain input (query, event, etc.).

Knowledge Base: It's a facts and rules collection. The facts are specified in a similar way as the relations in a relational database. The rules can be referred as "*situation-action*" or "*if-then*". The rules can generate a network of them according to the associations among them.

Reasoning Mechanisms: It's a reasoning process from the input data and the knowledge base. This mechanism is generic in the sense that it can be applied to different domains only by changing the knowledge base. The reasoning scheme can be deductive, inductive or abductive. The deductive reasoning can be from general to particular or from the premise to the logical conclusion. The abductive reasoning is a reasoning method used for general explication. The abduction starts with a conclusion and end derivating the conditions that could make valid this conclusion. The abduction tries to explain the conclusion. The inductive reasoning is the beginning from particular facts in order to reach a general conclusion [2, 3]. In figure 2 the ontological scheme of IDB concepts shown.

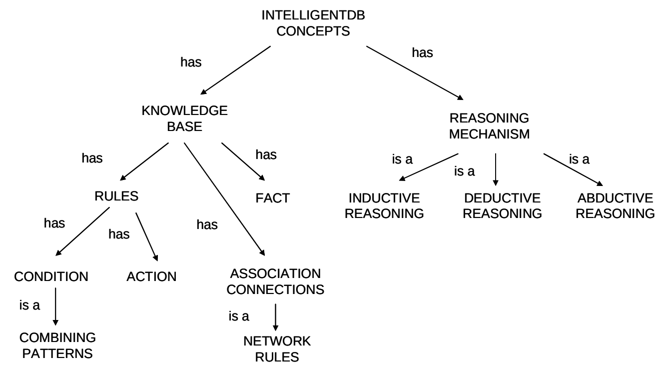

Figure 2. IDB Concepts for an Ontological Scheme

Next, table 2 shows the axioms for the IDB Concepts

TABLE II
AXIOMS FOR IDB CONCEPTS

Sentences	LPO
A IDB concept has a knowledge base and a reasoning mechanism	V x IDBConcept(x) => has(x,KnowledgeBase) Λ has(x,ReasoningMechanism)
A knowledge base has rules, has association connections and facts	V x KonwledgeBase(x) => has(x,Rules) Λ has(AssociationConnections) Λ has(x,Facts)
A rule has a condition, and has action	V x Rule(x) => has(x,Condition) Λ has(x,Action)
An AssociationConnections is a network rules	V x AssociationConnections (x) => is_a (x, NetworkRules)
A condition is a combination of facts that occur to activate Rule	V x Conditions(x) => is_a (x,CombiningFacts) Λ is_a (x,ActivationRules)
A reasoning mechanism is a deductive, inductive and abductive	V x ReasoningMechanism(x) => is_a(x,Deductive) V is_a(x,Inductive) V is_a(x,Abductive)
In the deductive reasoning the conclusion is obtained of the Assumptions	V x DeductiveReasoning(x) => is_a(x,ConclusionOfAssumptions)
In the inductive reasoning of the conclusions are obtained of the facts	V x InductiveReasoning(x) => is_a(x, ConclusionOfFacts)
The abductive reasoning tries to explain the conclusion	V x AbductiveReasoning(x) => is_a(x, ConclusionOfHypothesis)

B. Intelligent Database Operations

The IDB operations are made trough the reasoning machine, which controls the rules fired. The cycle starts with an event that can be a query or an update and ends when there are no applicable rules. The reasoning machine searches for the rules that fulfill the condition. Then, the rules execute the actions that could involve changes on the knowledge and environment database. There are different reasoning strategies, according to the type of reasoning that is used:

classically it could be *linking forward* or *linking backward* type. The linking forward type comes from facts to fulfill conditions and execute action (creating new facts). The linking backward comes from desirable states and tries to fulfill the necessary conditions to get to them [2].

The rules execution semantic depends on how to execute the rules [1]. There are three ways of execution: immediate, differed, and disconnect. Under the immediate way the rule is process as fast as possible, under the differed way the rule is process by the end of the transaction, under the disconnected way the rule is processed out of the transaction as a part of a separate transaction.

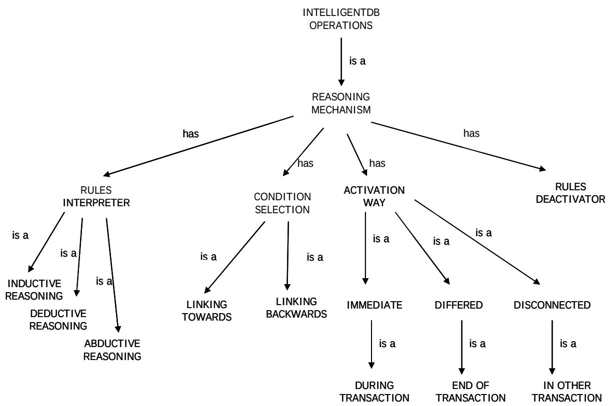

Figure 3. IDB Operational Ontological scheme

Next, on table 3 are shown the IDB Operational axioms.

TABLE III
IDB OPERATIONAL AXIOMS

SENTENCE	LPO
An IDB operation is a reasoning mechanism	V xIDBOperations(x) => is_a (x,ReasoningMechanism)
A Reasoning Mechanism is a rules interpreter, a rules executer, and a rules deactivator	V xReasoningMechanism(x) => is_a(x,RuleInterpreter) V is_a(x,RuleExecuter) V is_a(x,RuleDeactivator)
A rule interpreter is a deductive, inductive and abductive reasoning.	V x RuleInterpreter(x) => is_a(x,DeductiveReasoning) V is_a(x,InductiveReasoning) V is_a(x,AbductiveReasoning)
The rules executer has a condition selection and a activation way	V x RulesExecuter(x) => has(x,ConditionSelection) Λ has(x,ActivationWay)
The condition selection is a linking toward or linking backward	V x ConditionSelection(x) => is_a(x,LinkingToward) V is_a(x,LinkingBackward)
The activation way is immediate, differed or disconnected	V x ActivationWay(x) => is_a (x, Immediate) V is_a (x,Differed) V is_a(x,Disconnected)
The immediate activation way is the processing of the rule in transaction	V x Immediate(x) => is_a(x,ProcessingRuleinTransaction)
The differed activation way is the processing of the rule by the end of transaction	V x Differed(x) => is_a(x, ProcessingRulebytheEndOfTransaction)
The activation way of the disconnect rule is when the rule is process as another transaction	V x Disconnect(x) => is_a(x, ProcessingRuleInOtherTransaction)

C. *Intelligent Database Restrictions*

The IDB restrictions come according to the following conditions: a) If simultaneous firing rules arise, which is when an event or query has different associate rules and the system allows only one rule to activate. It can be solved by: random selection, use of priorities, establishing time activation of the rule, etc. b) If contradictions between rules arise, this is when an event or query firing two rules and each one generates an action which is the negation of the action generated by the other rule. In this case, that can be solved inhibiting the activation of some of them.

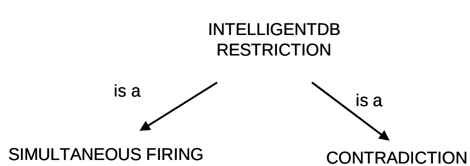

Figure 4. Ontological Scheme of IDB Restrictions

TABLE IV
AXIOMS OF THE IDB RESTRICTIONS

Sentence	LPO
The IDB restrictions occur for a simultaneous firing or contradiction between rules	V x IDBRestrictions(x) => is_a(x,SimultaneousFiringOfRules) V is_a(x,ContradictionBetweenRules)
In a simultaneous firing of rules a random selection of rules is made, the use of priorities, or fixed the activation time of the rule	V x SimultaneousFiringOfRules (x) => is_a(x,RandomSelectionofRules) V is_a(x,UseOfPriorities) V is_a(x,FixedActivationTime)
The contradiction between rules is solved inhibiting the rule activation	V x ContradictionBetweenRules (x) => is_a(x,InhibitingActivactiondeRule)

IV. CASE OF STUDY

To continue, a system intelligent of registration for a university is described, which is based on an IDB to manage the systems data. For these descriptions we will use the ontological frame proposed in this paper. We will call the system, Registration Intelligent System (RIS).

A. *General Description*

The RIS contains a knowledge base, which will be the IDB base, conformed by a fact base of students, and a rules base to make the students registration. Some examples of the information contained in them are: The **facts base** store students data, courses to attend, student's academic history, among others. The **rules base** stores rules that determinate the conditions in which can accept the student's registration in different courses that are offered. Some examples are:

a. Rule to establish the student register order, i.e.:

IF average student is superior or equal than 18 **THEN** register in the first established date.

IF average student is between 18 y 15, **THEN** register in the second established date.

b. Rules that allows the registration of the students according to an established status (new or regulars students, etc,)**.** For example:

IF student is regular **THEN** the credit numbers to inscribe is bigger than 12

IF the student is new **THEN** assign the first semester courses

c. Rules that establish the capacity of students in each course. For example:

IF computers have a laboratory **THEN** number of students=24

IF Analysis doesn't require a laboratory **THEN** number of students=45

d. Rules that allow the registration of courses according to the precedent among them. For example:

IF Course System Design approved **THEN** Register Language and Semantics

e. Exception Rules to register student, i.e.:

IF student last semester and ask parallel courses (courses with precedent relation among them) **THEN** accept parallel

f. Rules to open new courses, for example:

IF request for new course **THEN** verify if there is a professor

IF there is a professor **THEN** Check if there is a classroom

IF there is a classrooms **THEN** open new courses

Next, the IDB is described using our ontological framework.

B. Conceptual Description of the Intelligent Database using our ontological frame

Through the ontological framework for IDB, the RIS concepts and components are identified. The table 5 shows the use of our ontological framework in this case. It describes some of the conceptual components of the IDB's RIS as described in section 3.1 and figure 2. The Intelligent Database attributes are:

ID_IDB:DBI01

Name_IDB: Registration

Address: www.university.registrations

Domain: Academic

Scheme: a) Facts Base conform by: STUDENTS, CURRICULUM, CARREERS, GRADES, TEACHERS, CLASSROOMS, REGISTRATION, and Rules Base Rules Base, which contain the conditions under which we can authorize registration of students in the different courses offered, as well as managing the different situations (see previews examples of rules).

Model: Oriented Object Model is used to model schemes.

TABLE V
DBI CONCEPTUAL COMPONENTS SCHEME UNDER STUDY USING OUR ONTOLOGICAL CONCEPT SCHEME.

Concepts	LPO	Commentary
The IDB has a knowledge base and a reasoning mechanism	V x ConceptBDIntelligent(x) => has(KBRIS,KnowledgeBase) Λ has(RMRIS,ReasoningMechanism)	Description of System
The Knowledge Base has rules and facts. The **rules base** is conformed by: Rule to establish the student register order, Rules that allow registration of courses according to the precedent among them, Rules that establish the capacity of students in each course, Exception Rules to register student, Rules to open new courses, etc. The **fact base** is conformed by: student data, approved courses, courses to enroll, etc.	V x KnowledgeBase(x) => [has(x,RulesOrderRegistration) Λ has(x,RulesStatusStudents) Λ has(x,RulesOfCoursesOrder) Λ has(x,RulesExceptionRegister) Λ has(x,RulessCapacityOfCourses) Λ has (x, RulesCoursesPrecedent) Λ …] Λ [has(x, StudentData) Λ has(x,ApprovedCourses) Λ has(x, CoursesToRegister) Λ …]	DBI component description
Rules have a condition and action	V x Rule(x) => has(x,Conditión) Λ has(x,Action) For example: V x RuleOfOrderOfRegistration(x) => has(x,CONDITION(RegisterRequest ANDStudentselectbyAverage)) Λ has(x,ACTION(RegisterDate))	For de Rule "RuleOfOrderOfRegistration": has the EVENT firing Register Request with CONDITION Average Student to execute ACTION Register Student
Reasoning mechanism is deductive	V x ReasoningMechanism(x) => is_a(x,Deductive) For example: V x RequestRegister(x) => has (x, RulesRegisterOrder) Λ has (x, RulesCoursesPrecedent) Λ…]	The system deducing that to be doing. Happening Event "Register Request" to activate rules: that to establish "Register order by average of student" and "Courses of Programs"

C. Example of operations over the RIS

Following, we will describe examples of operation that can be made with the RIS, for which we use the ontological framework of section III B.

1) Student registration in a course

In this section we explain the student inscription in a given course. If the event that activates the knowledge base is student registration, RIS must verify if the student and the courses that the student wants to register exists, among other things. Then, the reasoning mechanism starts activating rules that allows making the registration.

TABLE VI
OPERATIONS IN THE RIS TO REGISTER A STUDENT

Operations	LPO	Commentary
Initial condition: Student Registration	V x ApplicationRegistration(x) => [is_a(x, Student) Λ is_a (x, ApprovedCourses t) Λ is_a (x, CoursesToRegister t] Λ…] Λ [is_a (x, RulesStatusStudent) Λ is_a (x, RulesCapacityCourses) Λ is_a (x, RulesOrderRegistration) Λ …]	V x Conditions(x) => is_a (x,CombiningFacts) Λ is_a (x,ActivationRules) (Axiom Table 2)
The Reasoning Machine of RIS interprets, executes and disabled rules	V x ReasoningMachineRIS(x) => has(x, InterpreterRulesRIS) Λ has (x, ExecutionRulesRIS) Λ has(x, DesableRulesRIS)	V xReasoningMechanism(x) => is_a(x,RuleInterpreter) V is_a(x,RuleExecuter) V is_a(x,RuleDeactivator) (Axiom Table 3)
The rules interpreter makes deductive reasoning	V x RulesInterpreterRIS (x) => is_a(x,DeductiveReasoning) For example: IF exist Application for Registration THEN activarte Rules of Register Order, StudentSatus, Courses Precedence, Classroom Capacity and exception rules	V x RuleInterpreter(x) => is_a(x,DeductiveReasoning) V is_a(x,InductiveReasoning) V is_a(x,AbductiveReasoning)) (Axiom Table 3)
Implementation Rule which initiates process of reasoning in RIS: IF Application for Registration THEN Selected Students by average AND … AND Courses Selected	V x ApplicationFor Registration (x) => is_a(x, LinkingToward) V x ApplicationFor Registration (x) => is_a (x, RulesRegisterOrder) Λ …Λ is_a(x,RulesSelectionCourses)	V x ConditionSelection(x) => is_a(x,LinkingToward) V is_a(x,LinkingBackward) (Axiom Table 3)
Example RulesRegisterOrder IF SelectionStudentand Average THEN Registration Dates Set	V x RulesExecuter (RulesOrderRegister) => has(x, ApplicationFor Registration) Λ has(x,ActivationWay) V x RegisterOrder(x) => has(x, CONDITION(SelectionStudentByAverage)) Λ has(x,ACTION(Set dates Registration))	V x RulesExecuter(x) => has(x,ConditionSelection) Λ has(x,ActivationWay) V x Rule(x) => has(x,Condition) Λ has(x,Action) (Axioms Tables 2 and 3)
Example of RulesSelectionCourses IF SelectionCourses THEN RegisterStudent	V x RuleExecute(SelectionCourses) => has(x, ApplicationFor Registration) Λ has(x, ActivationWay) V x SelectionCourses(x) => Λ has(x, CONDITION (Courses to Register, Courses Approved, Courses Precedence, Capacity Courses, status Student, etc.)) Λ has(x,ACTION (Make Registration))	V x RulesExecuter(x) => has(x,ConditionSelection) Λ has(x,ActivationWay) V x Rule(x) => has(x,Condition) Λ has(x,Action) (Axioms Tables 2, 3)
The activation Way of rules of RIS is immediate	V x ActivationWayRIS (x) => is_a(x,InmediateRulesOfRIS)	V x ActivationWay(x) => is_a (x, Immediate) V is_a (x,Differed) V is_a(x,Disconnected) (Axiom Table 3)

Other rules that must activate to inscribe the student are those that verify the available courses, the courses capacity, etc.

2) Opening a Course

This second operation is opening a course. To open a new course it is necessary to check that there is such course in the program. Specifically, we have a situation where two rules can be executed simultaneously, but we need to choose one of them. The sentences to be formulated to perform this operation are shown in the following table.

TABLE VII
OPENING COURSES

Operations	LPO	Commentary
Initial condition: Opening courses	V x OpeningCourses(x) => [is_a (x, Course) Λ is_a (x, Program)] Λ is_a (x,RulesAvailabilityProfessors) Λ is_a (x,RulesClassroom) Λ …)	V x Conditions(x) => is_a (x,CombiningFacts) Λ is_a (x,ActivationRules) (Axiom Table 2)
Implementation Rule which initiates process of reasoning in RIS IF OpeningCourse THEN Availibility Classroom AND … AND Availability Proffesors	V x OpeningCourses (x) => is_a(x, LinkingToward) V x OpeningCourses(x) => is_a (x,RulesAvailability Professors) Λ … Λ is_a (x,RulesClassroom)	V x ConditionSelection(x) => is_a(x,LinkingToward) V is_a(x,LinkingBackward) (Axiom Table 3)
Firing simultaneously rules We must prioritize between the two rules. In this case , we first need to verify the availability of professor, and then the classroom capacity	V x FiringSimultaneousRules(x) => is_a(x,RulesAvailabilityProfessors) Λ is_a (x,RulesClassroomCapacity)	V x SimultaneousFiringOfRules (x) => is_a(x,RandomSelectionofRules) V is_a(x,UseOfPriorities) V is_a(x,FixedActivationTime) (Axiom table 4)
IF Availability Professors THEN OpenCourse	V x RulesExecute(AvailibilityProfessor) => has(x, OpenCourse) Λ has(x,ActivationWay) V x ProfesorinCharge(x) => has(x, CONDITION(AvailibilityProfessor)) Λ has(x,ACTION(SetCourse))	V x RulesExecuter(x) => has(x,ConditionSelection) Λ has(x,ActivationWay) V x Rule(x) => has(x,Condition) Λ has(x,Action) (Axioms Tables 2 and 3)
IF AvailabilityClassroom THEN OpenCourse	V x RulesExecute(AvailabilityClassroom) => has(x, OpenCourse) Λ has(x,ActivationWay) V x Classroom (x) => has(x, CONDITION(AvailabilityClassroom)) Λ has(x,ACTION(OpeningCourse))	V x RulesExecuter(x) => has(x,ConditionSelection) Λ has(x,ActivationWay)) V x Rule(x) => has(x,Condition) Λ has(x,Action) (Axioms Tables 2 and 3)

D. Implementation in Protégé

Below, we show the ontology for IDB in Protégé OWL (DL) [11]. The figure 5 shows the taxonomy for IDB (left column). Here are defined the classes or concepts of the ontology of the IDB. We can see that the database concepts, operations and restrictions are subclasses of the IDB class. This class hierarchy is called taxonomy.

In OWL subclass means necessary implication. In other words, if **IntelligentDBConcepts** is a subclass of **IntelligentDB,** then all the instances (are called individual in Protégé) of **IntelligentDBConcepts** are instances of **IntelligentDB** without exception.

In the column that says properties and restrictions is shown the object properties that relate the concepts of IDB, as well as their domain and range properties. The domain and range property connects individuals of a domain with individuals of a range. For example, the relationship property ***hasIntelligentDBConcepts*** have domain **IntelligentDB** and range **IntelligentDBConcepts**.

The column that says superclass shows the superclass of **IntelligentDBConcepts (IntelligentDB)**. The classes **IntelligentDBConcepts**, **IntelligentDBOperations** and **IntelligentDBRestriction** are disjoint among them. This guarantees that an individual that is member of one of the classes in the group cannot be member of any other classes in that group.

All these properties and concepts define formally the ontology for the IDB (domain, behaviour, etc.) in order to allow inference processes over it.

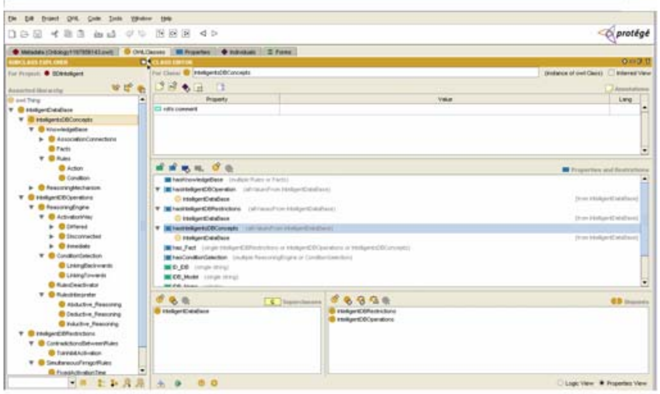

Figure 5. Representation of the concepts and properties in Protégé OWL (DL).

The figure 6 shows the individual (instance) of our ontology in the study case (the RIS).

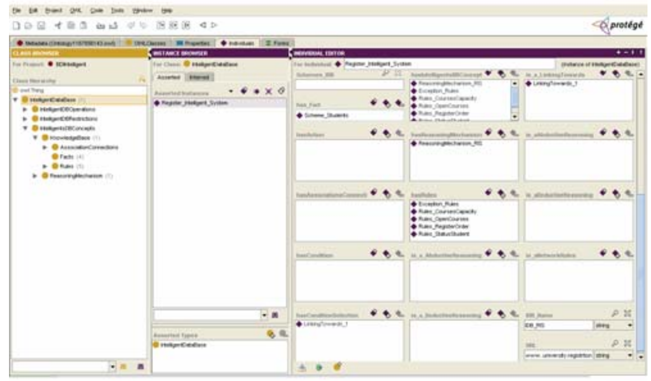

Figure 6. Individual of the IDB Ontology to describe the rule base for the RIS

The forms shown in figure 6 at the right half of the screenshot are generated automatically from the class definition. This figure shows an instance of the IDB Ontology, with the rules and facts presented in Table 5. The existence of this individual shows that the class is consistent and that the properties are well defined (Protégé has a verification procedure to guarantee that all individuals that have been defined through it, would be semantically correct).

V. CONCLUSION

In this paper we consider as IDB the active and deductive databases, and the knowledge based systems. We have presented the ontological schemes that represent IDB concepts, operations and restrictions, allowing the incorporation of reasoning mechanisms to the IDB.

We presented an example of utilization of our ontological framework, using the sentence of predicate of first order of it to describe a RIS. In addition, we explained the use of the RIS described using framework in two operations. The reasoning type used is the deductive, because from facts such as approved courses the system deduces the possible courses to register.

In addition, this paper shows an initial implementation of the IDB ontology in Protégé OWL. This implementation is not complete and upcoming research to make integration of databases will be made.

ACKNOWLEDGMENT

This work was supported by CDCHT-ULA grant I-820-05-02-AA.

REFERENCES

[1] Baral C., Lobo J.; "Formal Characterization of Active Databases", Lecture Notes in Computer Science; Vol. 1154, pp: 175 – 195, 1996
[2] Bertino E., Catania B., Zarri Gian P.; "Intelligent Database System", Addison-Wesley. 2001.
[3] Eberhart A; "Ontology-based Infrastructure for Intelligent Applications", CiteSeer.ISTcientific Literature Digital Library, 2004
[4] El-Helw Amr, Aly Hussien H; "An Intelligent Database Application for the Semantic Web"; CSITeA-04, Cairo - Egypt, 2004
[5] Gómez-Pérez Asunción, Fernández-López Mariano, Corcho Oscar; "Ontological Engineering", Springer-Verlag, 2004

[6] Muñoz A., Aguilar J.; “Architecture for Distributed Intelligent Databases”. IEEE, 13th Euromicro Conference on Parallel, Distributed and Network-based Processing, Euromicro-PDP 2005, pp 322-328
[7] Muñoz A, Aguilar J., Martinez R, "Integration Ontology for Distributed Database", Advanced Software Engineering: Expanding the Frontiers of Software Technology (Ed. S.Ochoa, G. Roman), IFIP, Springer, 2006 pp. 85-93
[8] Muñoz A, Aguilar J., Martinez R., "Modelo Inteligente para Bases de Datos Distribuidas", Revista Gerencia Tecnológica Informática (electronic journal: www.cidlisuis.org/aedo), Instituto Tecnológico Iberoamericano de Colombia, No. 10, Vol. 4, 2005 pp. 91-116,
[9] Ralston A, Reilly E, Encyclopedia of Computer Science and Engineering, Thomson Learning, 1993
[10] Van Heijst et al; "Using Explicit Ontologies in KBS Development" International Journal of Human and Computer Studies, 1996
[11] http://protege.stanford.edu/

Professor Ana Muñoz is a PhD student in the Computer Science Department at ULA Venezuela. She gained her MSc in Automation and Control in ULA and BSc in System Engineering from the ULA. He is an Assistant Professor in the Department of Technology at the Instituto Tecnológico de Ejido. Professor. Muñoz has been a visiting research/student in Murcia University in the ontology area. She has published papers in the field of ontology and knowledge management. Her research interests are in the areas of Ontology, Semantic Web, Federated Database and knowledge management**.**

Professor Aguilar Jose received the B. S. degree in System Engineering in 1987 (Universidad de Los Andes-Venezuela), the M. Sc. degree in Computer Sciences in 1991 (Universite Paul Sabatier-France), and the Ph. D degree in Computer Sciences in 1995 (Universite Rene Descartes-France). He was a Postdoctoral Research Fellow in the Department of Computer Sciences at the University of Houston (1999-2000). He is a Titular Professor in the Department of Computer Science at the Universidad de Los Andes. He has published more than 200 papers and 5 books, in the field of parallel and distributed systems, computational intelligence, science and technology management, etc. Dr. Aguilar has been a visiting research/professor in different universities and laboratories, has been the coordinator or inviting research in more than 20 research or industrial projects, and has supervised more than 20 M. S. and Doctoral students in their thesis.