Integrating Intelligent Methodological and Tutoring Assistance in a CASE Platform: The PANDORA Experience

Elena Castro   ***Dolores Cuadra***   ***Paloma Martinez***   ***Ana Iglesias***

ecastro@inf.uc3m.es **dcuadra@inf.uc3m.es** **pmf@inf.uc3m.es**   **aiglesia@inf.uc3m.es**

University Carlos III of Madrid, Spain

Abstract

Database Design discipline involves so different aspects as conceptual and logical modelling knowledge or domain understanding. That implies a great effort to carry out the real world abstraction task and represent it through a data model. CASE tools emerge in order to automating the database development process. These platforms try to help to the database designer in different database design phases. Nevertheless, this tools are frequently mere diagrammers and do not carry completely out the design methodology that they are supposed to support; furthermore, they do not offer intelligent methodological advice to novice designers.

This paper introduces the PANDORA tool (acronym of Platform for Database Development and Learning via Internet) that is being developed in a research project which tries to mitigate some of the deficiencies observed in several CASE tools, defining methods and techniques for database development which are useful for students and practitioners. Specifically, this work is focused on two PANDORA components: Conceptual Modelling and Learning Support subsystems.

Keywords: CASE tools, Database Design Methodologies, Intelligent Tutoring systems.

Introduction

Currently, there exist two main tendencies in the database design methodologies; one of them concerns those methodologies that are derived from Teorey, Yang & Fry (1986) which are calling relational databases design methodology. The second approach is related to the inclusion of object oriented data models in the database design (Rumbaugh, Blaha & Premerlani, 1991). Both of them have CASE tools which provide automated support to database design methodologies.

Although this paper does not concern object oriented database design, it is important to indicate that Unified Process (OMG, 2000) or IDEA (Ceri & Fraternali, 1996) are outstanding methodologies with commercial CASE support such as Rational Rose environment.

Database Design Methodologies (Teorey, Yang and Fry, 1986) generally use as conceptual data model the Entity Relationship (ER) model (Chen, 1976). Next methodological steps such as logical design, normalisation process and physical design, vary from some methodologies to others; for instance, in some methodologies it is supposed that if a good conceptual design is achieved, the normalisation phase is not necessary (Elmasri & Navathe, 2000, Silberschatz; Korth & Sudarshan, 2001).

Commercial CASE tools for database development do not usually cover database design phases with

real ER schemata, and they do not incorporate capabilities for refinement and validation processes and, in most cases, they manage hybrid models (merging aspects from ER and Relational models or using a subset of ER graphical notation for representing relational schemata) and sometimes these models are too close to physical aspects.

Next section is devoted to show the principal deficiencies in current CASE tools. Following, PANDORA project, whose main objective is to build a prototype of a CASE tool to be useful in database relational design and learning, is described focusing on its main contributions. Finally, some conclusions are presented.

Comparative Analysis Of Case Platforms

Constructs		BINARY MODELS		N-ARY MODELS	
		Designer 2000	ERWIN (IDEF1X)	Sylverrun	Power Designer
Entity Type					
	Regular	**YES**	**YES**	**YES**	**YES**
	Weak	No	Yes	Yes	Yes
Domains		No	No	Yes	No
Attribute Types					
	Main Identifier	Yes	Yes	Yes	Yes
	Alternate Identifier	Yes	Yes	Yes	Yes
	Derived	No	No	No	No
	Multivalued	No	No	Yes	No
	Composed	No	No	No	No
	Optional	Yes	Yes	Yes	Yes
	Foreign Key	No	Yes	No	No
Relationship Types					
	1:N Binary	**YES**	**YES**	**YES**	**YES**
	N:M Binary	Yes	Yes	**YES**	**YES**
	N-ary	**NO**	**NO**	**YES**	**YES**
	Cardinality Constraints				
	Max Cardinality	Look across	Look across	Look here	Look here
	Min Cardinality	Look here	Look across	Look here	Look here
	Attribute Relationship	**NO**	**NO**	**YES**	**YES**
	Generalization/Specialization	Complete and Disjoint	Complete or Incomplete and Disjoint	Complete and disjoint or Overlapping	Complete and Disjoint
	Existence	Yes	No	Yes	No
	Identifying	Yes	Yes	Yes	Yes
Methodological Assistant					
	Syntantic Validation	Yes	Yes	Yes	Yes
	Semantic Validation	No	No	Yes	Yes
	Assistant designer	No	No	No	No

Table 1: Comparative table

In order to analyse the capabilities and inadequacies of CASE technology, below a comparative study of four CASE tools (Designer2000, Erwin, Sylverrun and Power Designer because of their popularity) is presented. For each CASE tool, three features have been studied: ER constructs supported, ER schemata validation performed and, finally, if methodological assistance to facilitate modelling processed is provided. Table 1 shows a summary report.

The ER constructs analysed are:

1. *Entity Types:* All CASE tools studied draw regular entities, but weak entities only can be represented when there exist an identifying relationship.

2. *Relationship Types*: CASE tools can be classified in binary CASE and n-ary CASE tools depending on if they support binary or n-ary models (Song, Evans & Park, 1995). Binary CASE tools only represent binary relationships and, as they draw a line for each relationship, attributes in relationships can not be represented. Relating to cardinality constraints, the terminology *look across* and *look here* refers the place where the cardinality (maximum) or participation (minimum) constraints are specified in ER schemata (by looking across the relationship from the other direction or looking here first).

3. *Generalizations*: All CASE tools let users to represent generalisations but, in general, only complete and disjoint generalisations are allowed. Also they can represent identifying dependencies, but only two of them represent existence dependencies.

4. *Semantic constraints on attributes:* Main and alternative identifiers can be defined in all tools; nevertheless composed and derived attributes can not be declared. Another important aspect that we have considered in our study is the foreign key constraint. The foreign key constraint appears in Erwin CASE as an attribute property, and in some binary models it appears when a one-to-many relationship is established, so we understand that some CASE tools mix logical and conceptual designs.

5. *Syntactic and semantic validation*: CASE platforms should provide tools incorporating syntactic and semantic validation rules in order to be able to refine schemata and to help beginners to learn techniques for improving their designs (Bouzeghoub, Kedad & Métais, 2000).

   All CASE tools studied here have the following syntactic validations:

   ➢ The possibility of having different entity and relationship names (uniqueness).

   ➢ One attribute can not exist independently of its entity or relationship.

   ➢ A relationship is a binary association between two entities not necessarily different entities.

   ➢ A relationship can not participate in other relationship (there are not relationships of relationships).

   ➢ Cardinalities are defined as positive integer number intervals, so minimum cardinality must be always less than maximum cardinality.

   ➢ There not exist cycles in the generalisation hierarchies.

   Semantic validation is more complex due to the complexity associated to the domain knowledge (Batra & Antony, 1994, Batra & Zanakis, 1994). Silverrun CASE tool presents several types of semantic validation interacting with the user by some questions that help, for example, to choose a main identifier for an entity or to validate cardinality constraints in a relationship.

6. *Transformation into logical models*: Perhaps, the most important gap in most CASE environments is that all ER constraints are lost when transformation rules are applied. All CASE tools automatically apply some basic rules, (Teory, 1999), but they don't support other kind of rules in order to check, for example, the completeness and disjoint of a hierarchy. So they don't carry out an "intelligent transformation".

Concerning this topic, it would be necessary a methodological assistant that gives advice to beginners and practitioners in order to which learn and help them to achieve a good design.
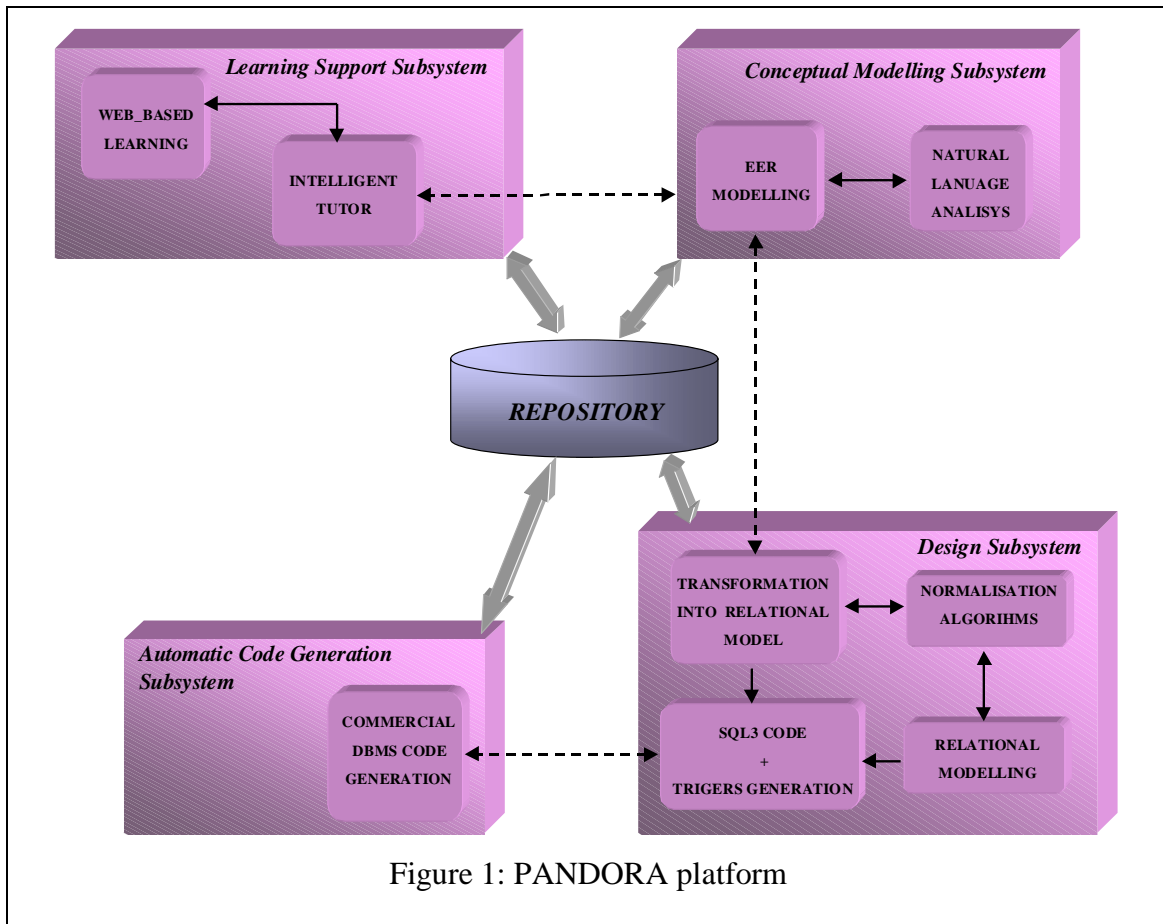
Figure 1: PANDORA platform

PANDORA Project

PANDORA (CASE Platform for Database development and learning via Internet. Spanish research CICYT project (TIC99-0215)) is a research project devoted to develop a CASE platform for database learning, design and implementation. It is composed of a set of modules (Figure 1) that can be independently used or in a methodological framework. In a first level, three layers are identified: conceptual modelling, design and automatic code generation subsystems. Over these layers, there is a learning support subsystem that provides an intelligent tutor for methodological assistance through the use of the different tools as well as a Web-based learning component.

The core of PANDORA platform is the *Repository* (metabase) that keeps all the resources and that supports the storage of Extended Entity-Relationship (EER) as well as relational schemata, SQL scripts, triggers and so forth. The design of the repository was decomposed in two metamodels, one for storing EER schemata and the other for storing relational schemata. Both of them describe all the constructs that they support. Moreover, this separation clearly distinguishes the two fundamental phases of the database development: conceptual and logical designs. Current CASE tools lack of this important distinction. Following, a brief description of PANDORA components along with their main contributions are given.

Conceptual Modelling Subsystem is composed of two modules: the EER Modelling and the Natural Language Analysis modules. The former is used by designers in drawing and verifying conceptual schemata; it also allows to store and retrieve conceptual schemata. The Natural Language Analysis module provides some facilities to interpreter a descriptive text in order to get proposals of EER schemata according to requirements appearing in the text, Martínez and García-Serrano (2000). Moreover, this module also supports an interactive process for identifying and validating binary relationship cardinalities in the conceptual modelling phase. This component profits from natural language processing techniques, first-order

logic and some modelling heuristics. Cardinality constraints, especially in higher order relationships, are difficult to understand and model by students, and some validation methods are required. What we propose is an approach that combines syntax (grammatical categories, word collocations, etc.), semantics (meanings of words, phrases and sentences) as well as first order logic to extract cardinality constraints and validate them with the user.

Concerning the *Design Subsystem*, it includes four modules: Transformation into Relational Model, Relational Modelling, Normalisation Algorithms and SQL-3 code (Melton & Simon, 2002) plus Triggers Generation modules. In order to achieve a transformation of EER schemata into the Relational model without loss of semantics, an exhaustive analysis of translating the different EER constructs has been performed. The aim is to develop databases that keep all the integrity constraints an that force their verification regardless of which program accesses the database.

In this subsystem, there are two main contributions: the first one is related to the repository, covering all elements proposed in SQL-3 (Melton & Simon, 2002), including the relational model constraints such as assertions, checks, primary keys, alternate keys and foreign keys constraints. Furthermore, inherent constraints are validated by triggers and checks. The second contribution concerns to the relational model transformation, converting the EER constructs into relational model's constructors preserving their associated semantics. A correct transformation of conceptual schemata and their associated constraints is necessary in order to preserve their intended meaning. Although relational model is insufficient for reflecting the complete semantics that can be presented in a conceptual schema, it can be enhanced with specific elements that are used to preserve the original semantics, such as active capabilities (triggers).

In the *Automatic Code Generation Subsystem*, the Commercial DBMS Code Generation module transforms the standard logical schema into an specific logical schema, taking into account the DBMS's characteristics and resolving the relational model's constraints.

Finally, the *Learning Support Subsystem* gives a coherent unification to the CASE environment from two perspectives. Firstly, the Intelligent Tutor module plays the role of a methodological assistant for guide the designer during the database development process (through the different phases) providing support in the design alternatives; the methodology for database development incorporated in PANDORA tool is explained in De Miguel, Piattini & Marcos (1999). Secondly, the Intelligent Tutor incorporates some teaching and training strategies of database design concepts that can be used via Web. For this purpose, a set of didactic units together with a set of exercises, have been designed.

In order to perform this pedagogic component we collaborate with people from the New Information Technologies and Communication Programme (PNTIC) belonging to the Spanish Ministry of Education, Culture and Sports that has a wide experience in designing and implementing Web courses for distance learning. They help us to define the didactic units and also they will provide us a platform to test the Learning Support subsystem with a national coverage.

Conceptual Modelling Subsystem

The main contributions in the Conceptual Modelling Subsystem are to include all EER model constructs such as regular and weak entities, high degree relationships, minimum and maximum cardinality constraints, attributes in relationships, optional and mandatory attributes, monovalued and multivalued attributes, single and composite attributes, partial and complete hierarchies, overlapping and disjoint hierarchies, existence and identifying relationships. The repository has to permit storing schemata containing all these elements. It supposes not only to allow drawing these constructs but to consider them in the verification of schemata (to check if conceptual schemata are non redundant, consistent and complete) (Bouzeghoub, Kedad & Métais, 2000), and in transforming conceptual schemata into relational model in
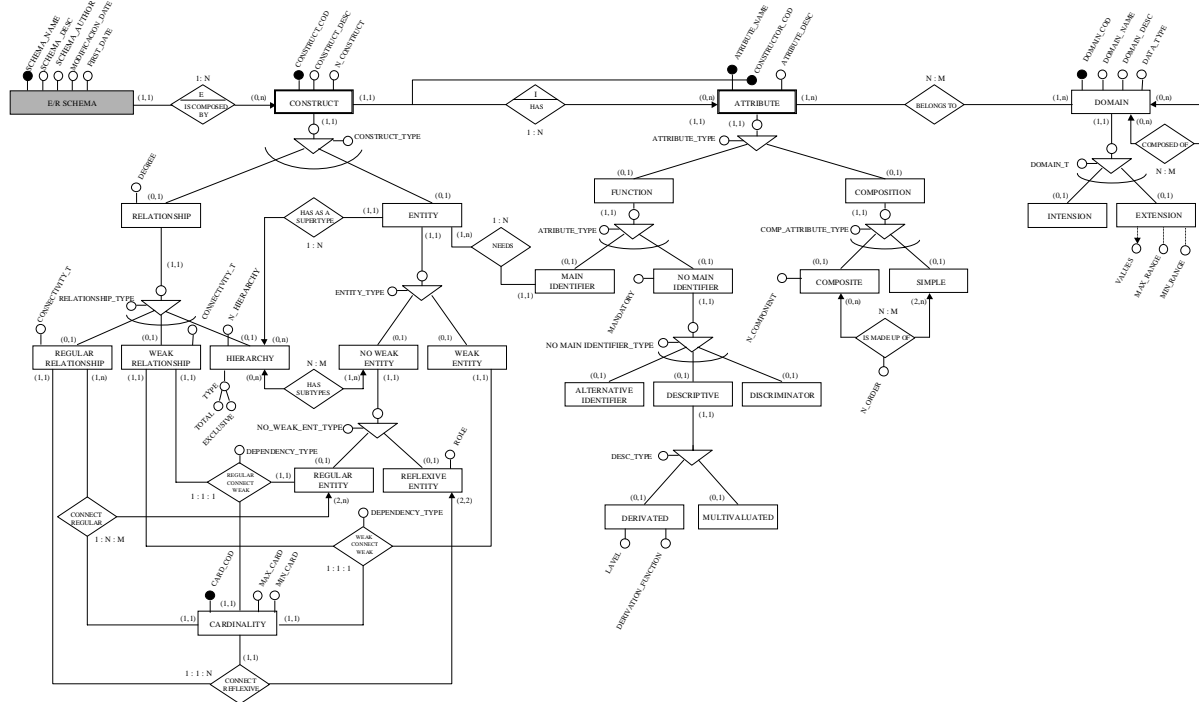
Figure 2: Conceptual schema of repository for EER schemata (metaschema)

the Design Subsystem. Figure 2 shows the conceptual schema of repository for EER schemata (meta-schema).

This metaschema collects all those aspects commented in previous sections. As conceptual models are not supported in a DBMS, the repository of the figure 2 must be transformed to the relational model in order to be implemented (Cuadra et al., 2002, Fahrner & Vossen, 1995). The DBMS used for this task has been Oracle 8i. Besides, some procedures and triggers has been implemented to check semantic validation.

Learning Support Subsystem

We are developing an Intelligent Tutor System (ITS), known as *RLITS* (Reinforcement Learning in ITS), to assist learners through the use of the different tools in this project as well as to methodologically teach them how to design and develop databases through the Web (Iglesias et al., 2002).

An ITS is defined as "computer-aided instructional systems with models of instructional content that specify *what* to teach, and teaching strategies that specify *how* to teach" (Wenger, 1987). *RLIT* System decides (by learning) not only what and how (text, image, video, animation, etc.) to show system's content (problems, exercises, tests, simulations) to current student, but *where* to do it, based only in the previous experience with others students with similar learning characteristics.

A typical structure of an ITS is composed of four well differentiated modules (Burns & Capps, 1988). The *student module* contains all necessary information about the student in the learning process: student knowledge, personal characteristics, historical behaviour, etc. The *interface module* facilitates the com-munication between the ITS and the student. The *domain module* includes all the characteristics about the knowledge to teach. The traditional hierarchical knowledge structure (topics, sub-topics, etc.) is used to define the tutor system's objectives. Figure 3 shows a proposal of a hierarchical structure for database
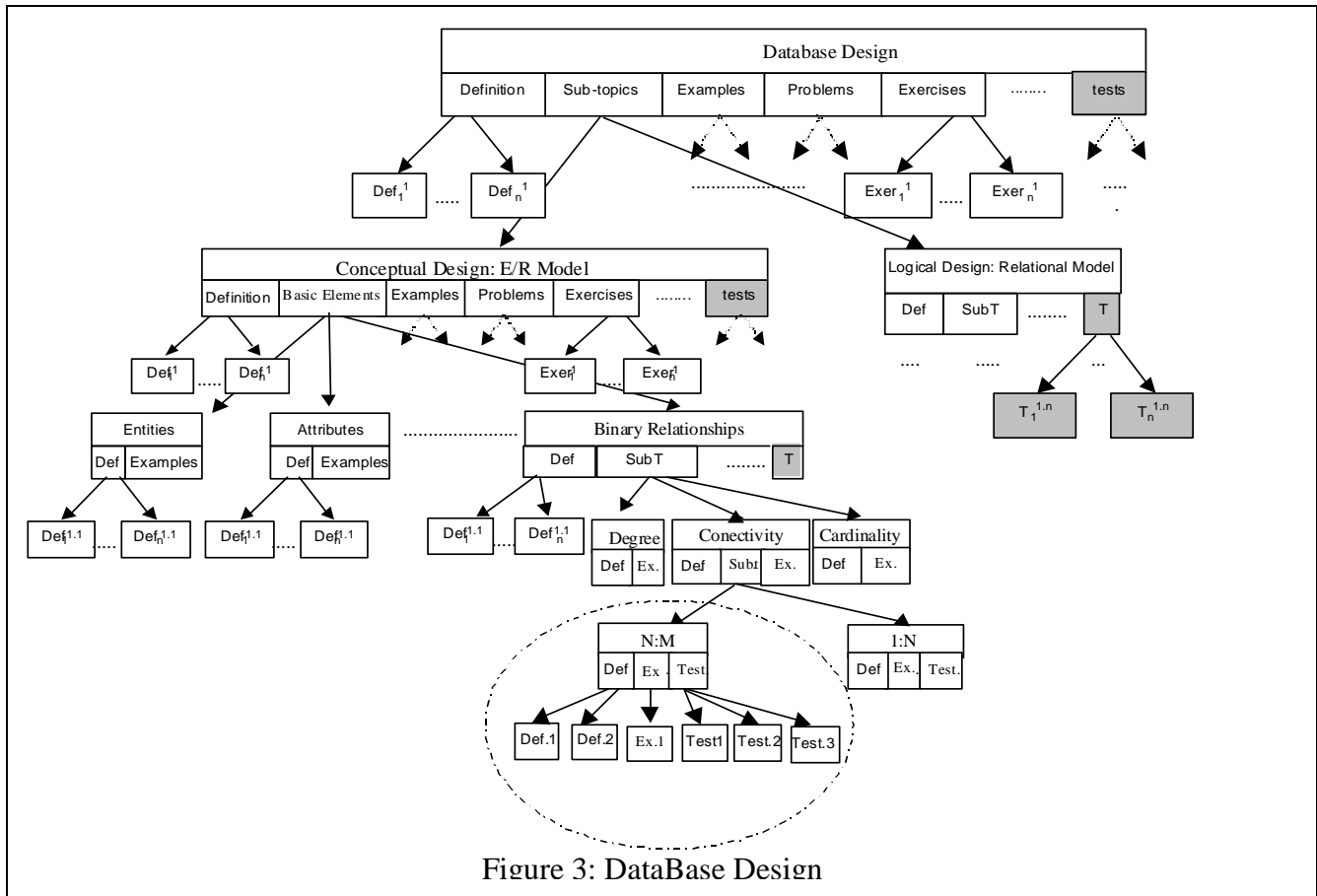
Figure 3: DataBase Design

design knowledge, where each topic has been divided into sub-topics, and these in others sub-topics, and so on. At the same time, each node of the tree contains sets of definitions, examples, problems, exercises, etc. in several formats (image, text, video, etc.). Finally, the *pedagogical module* decides *what*, *how* and *when* to teach the domain module contents, taking the better pedagogical decisions according to the student needs, that is, the ITS finds the best way to teach the *knowledge items,* corresponding with the internal nodes of the knowledge tree (topics), to the current student.

The definition of this problem as a reinforcement learning problem is fulfilled as follow (Iglesias, Martínez & Fernández, 2002): the agent's *state* is the current student's knowledge, represented by a vector of representative values of the student's knowledge for each topic. The ITS perceives the current student's knowledge (s_1) by evaluations (*tests).* Given one state, the system chooses an action to be executed according to the current *action policy*, *B* (Kaelbling, Littman & Moore, 1996). The *action* corresponds with showing leaves of the knowledge tree (*definition, exercise, problem*, etc.). This action is supposed to change the current state of the student's knowledge to a new state (s_2), generating a *state transition* and a *reinforcement signal* (positive or negative) to the system. This signal is used to update the system's action policy. The system behaviour, *B*, should choose actions that tend to maximise the long-run sum of values of the reinforcement signal, choosing in this way the optimal tutoring strategy (what, when, and how; the best sequence of contents and how to teach them) to coach the current learner by trial and error, like an human tutor does.

Conclusions

The two main aspects of PANDORA related in this paper are:

➢ The inclusion of the elements necessary to provide support to the relational databases methodology, taking into account deficiencies observed in several representative CASE tools.

➢ The use of a CASE tool as support of the database design teaching. Use this kind of tools may help designers in the most complicated design steps, carrying out the validation and refinement tasks as well as interacting with users. In addition, to assists learners through the use of the different tools in this project as well as to methodologically teach them how to design and develop databases through the Web.

Currently, we are at a prototype stage in which the repository has been implemented in Oracle 8 together with the EER and Relational diagrammers with their complete graphical features as well as associated functionalities. The prototype is implemented in Microsoft Visual Basic 6.

We are working in the Transformation into Relational model, implementing the translation rules for all EER constructs (Al-Jumaily, Cuadra & Martínez, 2002).

Relating the Natural Language Analysis module, the interpreter for cardinality constraints is also implemented in Prolog using the DCG (Definite Clause Grammars) formalism (Martínez et al., 2000). This component is calling by the Intelligent Tutor to give support in database design learning to teach the cardinality constraint concept.

References

Al-Jumaily, Cuadra & Martínez, (2002). An execution model for preserving cardinality constraints in the relational model. Proc. 4th International Conference on Enterprise Information Systems (ICEIS'2002), Ciudad Real, Spain.

Batra, D. and Antony S. R (1994). Novice errors in conceptual database design. *European Journal of Information Systems*, Vol. 3, N. 1, pp. 57-69, 1994.

Batra, D. and Zanakis, H (1994). A conceptual database design approach based on rules and heuristics. *European Journal of Information Systems*, Vol. 3, N. 3, pp. 228-239, 1994.

Bouzeghoub, M., Kedad, Z. & Métais E (2000). *CASE Tools: Computer Support for Conceptual Modelling*. In Advanced Database Technology and Design, Díaz and Piattini (Eds.), Artech House, 2000.

Burns, H. & Capps, C. (1988). *Foundations of Intelligent Tutoring Systems: An Introduction. Foundations of Intelligent Tutoring Systems*. Hillsdale, N.J: Lawrence Erlbaum Associates.1-19.

Ceri, S. & Fraternali, P. (1997). *Designing Database Applications with Objects an Rules: The IDEA Methodology*. Ed. Addison–Wesley, 1997.

Chen, P. P. (1976). The Entity-Relationship Model: Toward a Unified View of Data. *ACM Transactions on Database Systems*. 1, 1, 9-36.

Cuadra et al., (2002). *Preserving relationship cardinality constraints in relational schemata*. Database Integrity: Challenges and Solutions, Ed: Idea Group Publishing, 2002.

De Miguel, A., Piattini, M. & Marcos, E. (1999). Diseño de Base de Datos Relacionales. Ed. RAMA, 1999.

Elmasri, R. & Navathe, S.B. (2000). Fundamentals of Database Systems. Third Edition. Addison-Wesley.

Fahrner, C. & Vossen, G (1995). A survey of database design transformations based on the Entity-Relationship model. *Data & Knowledge Engineering*, 15, 213-250, 1995

Iglesias, A., Martínez, P. & Fernández, F. (2002). Applying Reinforcement Learning in Intelligent Tutoring Systems. Proc. 4th International Conference on New Educational Environments (ICNEE'2002), Lugano, Switzerland.

Iglesias et al. (2002). Learning to Teach Database Design by Trial and Error. Proc. 4th International Conference on Enterprise Information Systems (ICEIS'2002), Ciudad Real, Spain.

Kaelbling, L. P., Littman, M. L. & Moore, A. W. (1996). Reinforcement learning: A survey. *Int. J. of. Artificial Intelligence Research*, 237-285.

Martínez, P. & García-Serrano, A. (2000). On the Automatization of Database Conceptual Modelling through Linguistic Engineering. *Natural Language Processing and Information Systems (NLDB 2000). Revised papers. LNCS 1959,* pp. 276-287.

Martínez, P. et. al. (2000). Data conceptual modeling through Natural Language: Identification and Validation of Relationship Cardinalities. Proc. 11th International Conference of the Information Resources Management Association, Alaska.

Melton, J. & Simon, A. R. (2002). *SQL: 1999. Understanding Relational Language Components*. Morgan Kaufmann Publishers.

OMG (2000). Unified Modelling Language Specification, Version 1-3. Object Management Group. *ACM Computing Surveys* 31(1): 63-103.

Quinlan, J.R. (1993). *C4.5 Programs for machine learning.* Morgan Kaufmann Publishers, San Mateo, California.

Rumbaugh, J., Blaha, M. & Premerlani, W. J. (1991). *Object Oriented Modelling and Design*. Prentice Hall, Englewood Cliffs, New Jersey, 1991.

Silberschatz, A., Korth, F. & Sudarshan, S. (2001). *Database Design Concepts*. 4th ed. McGraw-Hill, July 2001.

Sison, R. & Shimura, M. (1998). Student Modelling and Machine Learning. *International Journal of Artificial Intelligence in Education*, 9, 128-158.

Song, I.Y., Evans, M. & Park, E.K (1995). A Comparative Analysis of Entity-Relationship Diagrams. *Journal of Computer & Software Engineering*, 3 (4), 427-459.

Teorey, T. J. (1999). *Database Modeling and Design: The Entity-Relationship approach*. 3rd ed. Morgan Kaufmann, San Mateo, 1999.

Teorey, T. J., Yang, D. & Fry, J. P. (1986). A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model. *ACM Computing Survey*. Vol 18. No. 2. June 1986.

Wenger, E. (1987). *Artificial Intelligence and Tutoring Systems*. Los Altos, CA: Morgan Kaufmann.

Biographies

Elena Castro received the M. Sc. in Mathematics from Universidad Complutense of Madrid in 1995. Since 1998 she works as assistant lecturer at the Advanced Databases Group in the Computer Science Department of Universidad Carlos III of Madrid. She is currently teaching Relational Databases. Her research interests include database conceptual and logical modelling, Advanced Database CASE environments and Information Engineering.

Dolores Cuadra received the M. Sc. in Mathematics from Universidad Complutense of Madrid in 1995. Since 1997 she works as assistant lecturer at the Advanced Databases Group in the Computer Science Department of Universidad Carlos III of Madrid. She is currently teaching Database Design and Relational Databases. Her research interests include database conceptual and logical modelling and Advanced Database CASE environments.

Paloma Martínez Fernández got a degree in Computer Science from Universidad Politécnica of Madrid in 1992. Since 1992, she has been working at the Advanced Databases Group in the Computer Science Department at Universidad Carlos III of Madrid. In 1998 she obtained the Ph.D. degree in Computer Science from Universidad Politécnica of Madrid. She is currently teaching Database Design, Advanced Databases in the Computer Science Department at the Universidad Carlos III de Madrid. She has been working in several European and National research projects about Natural Language Processing, Advanced Database Technologies, knowledge-based systems and Software Engineering.

Ana Iglesias got a degree in Computer Science from Universidad Carlos III of Madrid in 1999. She works as assistant lecturer and teacher at the Advanced Databases Group in the Computer Science Department of Universidad Carlos III of Madrid. Her research interests include conceptual modelling theories as well as Intelligent Tutoring Systems.