An Automatic Correction Tool For Relational Database Schemas

Ferran Prados, Imma Boada, Josep Soler and Jordi Poch
University of Girona, Computer Science Department,
Campus de Montilivi, 17071 Girona (Spain)
{ferran.prados, imma.boada, josep.soler, jordi.poch}@udg.es

Abstract **- A web-based tool developed to automatically correct relational database schemas is presented. This tool has been integrated into a more general e-learning platform and is used to reinforce teaching and learning on database courses. This platform assigns to each student a set of database problems selected from a common repository. The student has to design a relational database schema and enter it into the system through a user friendly interface specifically designed for it. The correction tool corrects the design and shows detected errors. The student has the chance to correct them and send a new solution. These steps can be repeated as many times as required until a correct solution is obtained. Currently this system is being used in different introductory database courses at the University of Girona with very promising results.**

Index Terms - Automatic Correction, Relational Database schemas, e-learning environments.

INTRODUCTION

To improve both teaching and learning at the technical/engineering degrees at the Girona University the Department of Mathematics and Computer Science of this University has been developing an e-learning platform. Besides the functionalities of a common e-learning platform our environment includes the following main features:

- *Support automatic generation and correction of problems*. In most of engineering courses there is a lot of practice. Students have acquired the theoretic concepts when they are able to solve the problems related with these. Due to the importance of practice the platform integrates the modules required to automatically generate and correct exercises. To generate different exercises, an exercise with a set of variable parameters and a list of possible values for these parameters, denoted base exercise, is entered in the system. The exercise generation module automatically creates as many different versions of the base exercise as the combination of these parameters allows. To carry out correction a solving method for each base exercise is also entered into the platform. When one of the generated exercises has to be corrected the correction module adjusts the solving method by taking into account the values of its variable parameters.
- *Provide students with a friendly scenario to solve practical problems.* The possibility to automatically generate and correct different exercises allows us to give personalized attention to each student. This means exercises specifically designed for him and continuous feedback from the platform. Each student has assigned a set of problems. When the student enters a solution to a problem the correction module corrects it on-line and shows detected errors. In case of errors he has the chance to correct them and send a new solution. Such an environment makes student feel comfortable and supported.
- *Support continuous assessment.* All the student solutions are stored in the data base of the platform providing teachers the information required to perform continuous assessment. Moreover, teachers can track student progress on the subject and detect their weaknesses.

The automatic generation and correction modules are the kernel of our e-learning platform. The key of both modules is on the definition of the base exercise which varies according to the subject . Currently the platform supports mathematics, physics and programming problems amongst others and is used in several subjects of the technical/engineering degrees of the Girona University with very promising results [2,3].

In this paper we present the module that has been designed to support the automatic correction of relational database schemas. Database design is a common subject in undergraduate database courses. In these courses students must be taken through all stages of database development: analysis, conceptual, logical and physical design. At the end of the course the student, amongst others things, has to be able to design relational database schemas. Due to the importance of this topic and motivated by the good results obtained with the platform, we decided to develop the module to support them.

The paper has been structured as follows. In Section 2 main topics related to relational database schemas design and an example of a database problem are given. In Section 3 we describe how a base problem has to be defined. The solving method applied by to correct these exercises is presented in Section 4. In Section 5 experimental results are reported. Finally, conclusions and future work are given in Section 5.

RELATIONAL DATABASE SCHEMAS

In this section we briefly review main concepts related to relational database schemas. We also present an example to show how a relational database schema is.

A database is a collection of related data representing some aspect of the real world. Designing a database involves specifying the data types, structures and constraints for the data to be stored in the database.

The database schema is the description of the database and it has to be specified during the database design. The relational data model is a method to represent the database as a collection of relations which in general resembles a table of values.

To illustrate these concepts we present an example of a relational database schema designed for a company [1]. The main requirements of this company are:

-The company is organized into departments. Each department has a unique name (dname), a unique number (dnumber), and a particular employee who manages the department (mgrssn). We keep track of the start date when that employee began managing the department (mgrstartdate).

-A department may have several locations (dlocation). A department controls a number of projects, each of which has a unique name (pname), a unique number (pnumber), and a single location(plocation).

-We store each employee's name (fname)(lname), social security number (ssn), address (address), salary (salary), sex (sex) and birth date(bdate). An employee is assigned to one department but may work on several projects, which are not necessarily controlled by the same department. We keep track of the number of hours (hours) per week that an employee works on each project. We also keep track of the direct supervisor (superssn) of each employee.

- We want to keep track of the dependents of each employee for insurance purposes. We keep each dependent's first name (depname), sex (depsex), birth date (depbdate), and relationship (deprelationship) to the employee.

Once the requirements of the database are known, the relational base schema aims to collect all these information in a set of tables. Since each table has assigned a set of attributes, to define it we have to determine which these attribute are. We also have to define the primary key, which is different for each table entry, and the foreign keys, which represent the relation between the different tables of the database. There are different methodologies to determine how these tables have to be defined. In the scope of this paper this phase is not of our interest. We are going to consider only the relational database schema i.e. the set of tables, since this is the part of database problem we want to automatically correct.

Following the example, the different attributes that have to be taken into account for the database design correspond to the words written in brackets. The relational database schema groups them in the set of tables given below, where the bold underlined attribute is the primary key and the attributes in italics are the foreign keys.

EMPLOYEE

ssn	fname	lname	Sex
bdate	address	salary	*superssn*
dnumber			

DEPARTMENT

dnumber	dname	mgrstartdate	*mgrssn*

WORKS_ON

ssn	***pnumber***	hours

DEPT_LOCATIONS

dnumber	**dlocation**

PROJECT

pname	**pnumber**	plocation	*dnumber*

DEPENDENT

Ssn	**depname**	depsex	depbdate	deprelationship

DATA BASE EXERCISES

In this section we describe how database exercises have to be defined in order to be supported by our e-learning platform.

Crucial to define the base exercise is entering the problem descriptor and one or more correct solutions of the problem, which will be used by the correction module. Therefore to define this new typology of exercises we have to give the specification of both items.

I. Problem Descriptor

One of the most attractive features of our e-learning platform is its capability to generate different versions of a base exercise. In the case of database problems our interest is not on the generation of different versions of the problem but in the automatic correction. We consider that the degree of variability of these problems is inherent to the problem descriptor and hence the automatic generation of new versions is not necessary.

To define the problem descriptor item the next consideration have been taken into account. To design a relational database schema of a problem the student has to define the tables and the different attributes that compose them. Therefore, essential to solve the problem is the identification of the different attributes. From the example of the previous section it can be seen that the problem descriptor is just a set of requirements of some aspect of a real world situation and hence, it is from these requirements where the student has to determine the attributes. For our purposes, we consider that more important than the capability to identify attributes is the capability to properly group them in the

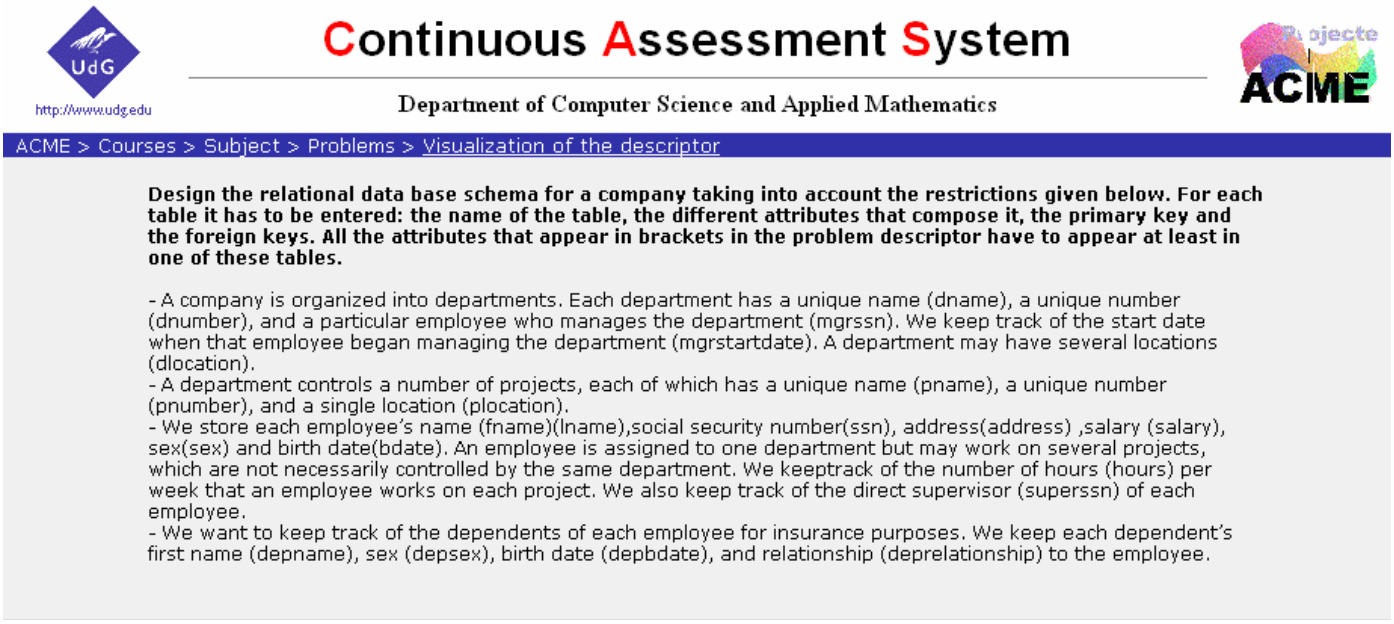

FIGURE 1.
STUDENT'S INTERFACE WITH A DATABASE PROBLEM. ATTRIBUTES ARE IN BRACKETS

correct tables. With all these considerations in mind we decided to fix the names of all attributes in the problem descriptor. Moreover, in this manner, as it will be seen in the next section, the correction of the schema will be easier.

The problem descriptor of a database problem consists of a set of requirement for a real life application where the attributes that have to be recorded in the database are represented as words in brackets. In Figure 1 we show the student's interface of the e-learning platform with the database problem to be solved.

II. Correct Solutions

The next item of the base problem we have to enter are correct solutions which will be used by the corrector module of the platform to correct the student's relational schema design.

The solution of a data base exercise is not unique. For this reason in the base problem we maintain some of these solutions. Each one of them consists of a set of tables with the corresponding attributes including primary and foreign keys, that satisfies all the requirements of the problem descriptor. Following the example of the previous section we maintain the solutions presented below. Observe that the first name is the name of the table and then three groups of attributes are given

```
Solution 1

Table  EMP
Primary key :       ssn
Foreign key :       dnumber superssn
Other attributes :  fname lname bdate address
                    salary sex

Table  DEP
Primary Key :       dnumber
Foreign key :       mgrssn
Other attributes : dname mgrstartdate


Table  WORKS_ON
Primary key :       ssn pnumber
Foreign key :       ssn pnumber
Other attributes :  hours


Table  LOCATION
Primary Key :       dnumber dlocation
Foreign key :       dnumber
Other attributes :

Table  PROJECT
Primary key :       pnumber
Foreign key :       dnumber
Other attributes :  pname plocation

Table  DEPENDENT
Primary key :       ssn depname
Foreign key :       ssn
Other attributes : depsex depbdate
                   deprelationship
```

AUTOMATIC CORRECTION OF DATA BASE RELATIONAL SCHEMAS

To describe how the correction module proceeds we have to consider first, how the student enters the solution into the system and second the correction method applied by the correction module.

I. Student Solution

We suppose that the student has designed a solution, he has identified all the attributes and he has determined the different tables that have to be defined. To enter this information in the system he uses the interface presented in Figure 2. At the bottom of this interface, from left to right, the main items are:

- *Table name*: this is a text box used to enter the name of the table. There are no restrictions about this name.
- *Field Name*: it is a list box with all the possible attributes that have appeared in the problem descriptor. The student has to select one or more of them.
- *Attribute Category*: it is used to select the category of the attribute which can be primary key, foreign key or a normal attribute (i.e. not a key)

Observe that in the interface there is also a graphical representation of the different tables that have been entered by the student. In the figure we can see the tables `works`, `employee` and `project` with the different attributes entered for the moment. Each time the student press the *Add* button of this interface the information of the items appears in the corresponding table. The information of the tables can also be modified. For example if we want to remove the field salary of the `employee` table we only have to click on it and delete. Once the student has entered the solution he presses the *Correct* button and the correction module applies the correction strategy. This strategy is described in next section.

II. Correction Method

The correction strategy is based on a complex comparison strategy which takes into account several parameters. The basic idea is to compare the student's solution with the possible correct solutions entered with the base problem. This comparison process is repeated until a complete coincidence between a correct and the student solution is found. If any one of the correct solution coincides with the one of the student the solution is considered incorrect. In this case the student receives information about the mistakes. Possible examples of error message are: *"More tables are required"*, *"You have to reduce the number of tables"*, *"The table XXX is not correct"*, etc. With these messages the student has an idea of where is the error and hence he can correct it and send a new solution.

Let us analyze how the comparison process proceeds. Note that any restriction on the names of the tables has been fixed,

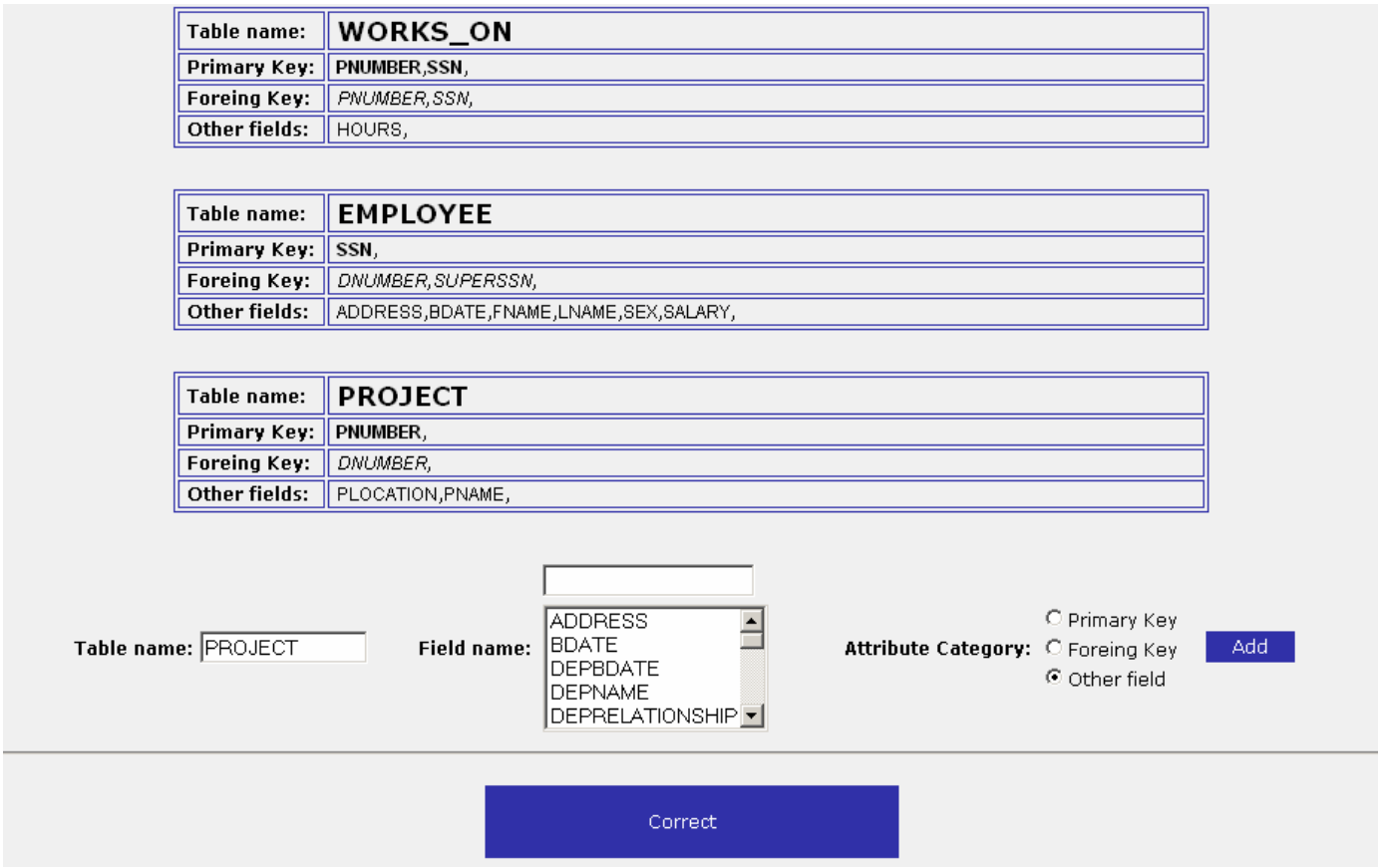

FIGURE 2.
STUDENT'S INTERFACE FOR ENTERING HIS SOLUTION

therefore to make the comparison we are not going to consider them. The key of the correction is on the names of the attributes and its category (primary key, foreign key or others). In a very high level the correction strategy can be seen as a process composed of the following four phases:

- *Phase 1. Compare number of tables.*

On a first step, the process compares if the number of tables entered by the student at least is equal to the number of tables of one of the correct solutions. In case that more than one solution coincidence exists all of them have to be evaluated until the perfect matching is found. When a coincidence is found the second phase of the process starts, otherwise an error message appears on the screen.

- *Phase 2. Compare primary keys.*

We compare the primary keys of the student's tables with the solution selected in the previous phase. If there is one solution with the same distribution of primary keys we start the next phase processing this solution. If there is an error in this phase the process is applied to another of the solutions previously selected. In case that none of the solutions matches the corresponding error message appears on the screen.

- *Phase 3. Compare foreign keys.*

As in the previous case there has to be a coincidence between all the foreign keys. If there is not, a message appears, and otherwise we apply the last phase. All the possible solutions are tested.

- *Phase 4. Compare other attributes.*

Finally, the attributes classified as others are compared proceeding as in the previous phases.

If all the phases are passed with success the student's solution is correct. Note that the matching process required to perform all the comparison is quite complex because there are a lot of possible combinations of the attributes and also the order in which the tables are entered. Moreover, the process always maintains all the correct solutions candidates to be matched with the student solution.

Two correction examples are illustrated in Figures 3 and 4, respectively. In the first case, see Figure 3, the student solution is composed of three tables. This solution is not correct. As it can be seen in the message window the design requires more tables, the entered tables DEPARTMENT and EMPL are not correct and there is also an incoherent field. In Figure 4 the student solution is correct. Note that there is a perfect matching with the solution of the base problem presented in the previous section.

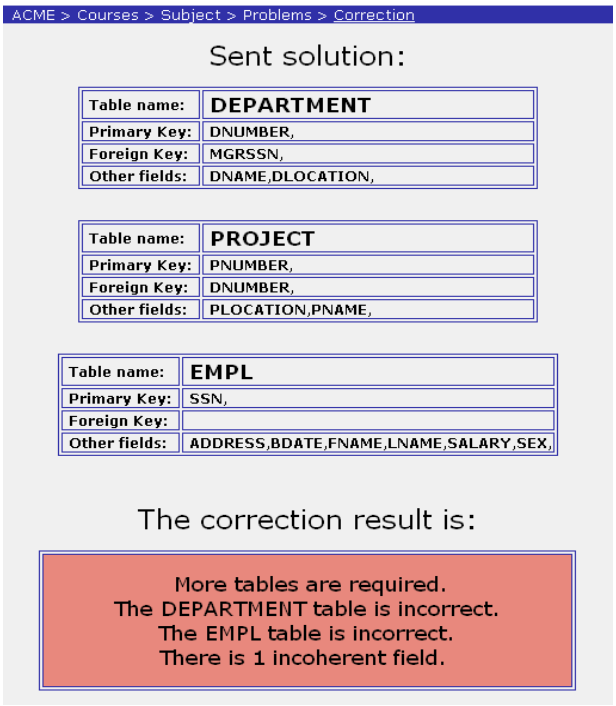

FIGURE 3. AN ERROR MESSAGE EXAMPLE

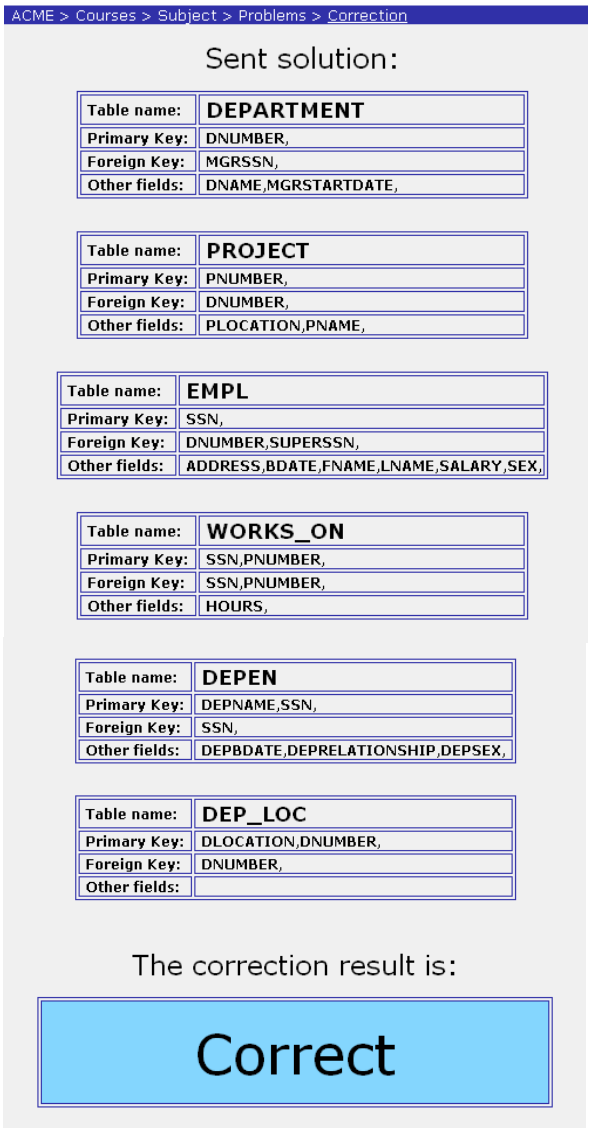

FIGURE 4. A CORRECT SOLUTION EXAMPLE

EXPERIMENTAL RESULTS

The presented module has been integrated in our e-learning platform. Currently, it is being used in two groups of 100 and 150 students, respectively. It is used as reinforcement in teaching and learning of introductory database courses.

From the teacher's first impressions, we can remark that the environment is easy to use. It does not require any installation, only a web connection. More importantly, it provides gains with respect to the classical teaching methodology in the sense that it offers a system for the continuous assessment of the student's progress, makes personalized attention to the student easier and assesses the degree of participation of the students.

The students' impressions have also been positive especially the fact that to access the system they only need an Internet connection. During the different sessions students were asked to comment on the problems they faced while using the system. The responses were very positive. The students feel motivated to solve the proposed problems. The possibility to correct a problem in real time encourages them to work until the correct solution is found.

An demonstration of our e-learning platform is available at *http://acme.udg.es/demoITHET05*.

CONCLUSIONS

We have presented a tool to automatically correct relational database schemas typical of database courses. This tool has been integrated in a more general e-learning platform and is used to reinforce teaching and learning of database courses.

Currently it is used in different courses with very promising results.

Our future work will be focused on the design and implementation of the modules required to correct SQL exercises and Entity-Relationship models.

ACKNOWLEDGMENT

ACME is supported in part by Education and Science Ministry from the Spanish Government EA2005-0094.

REFERENCES

[1] Elmasri R., Navathe B. "Fundamentals of DataBase Systems" *3rd edition. World Student Series, Addison-Wesley, Reading*, MA, 2000.

[2] Soler J., Poch J., Barrabés E., Juher D., Ripoll J., " A tool for the continuous assessment and improvement of the student's skills in a mathematics course", *Proceedings of the International Conference Technology of Information and Communication in Education for Engineering and Industry*, TICE 2002, 105-110.

[3] Boada I., Soler J., Prados F, Poch J., " A teaching/learning support tool for introductory programming courses", *Proceedings of the FIfth International Conference on Information Technology Based Higher Education and Training*, ITHET 2004, 604-609.