World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:6, No:4, 2012

Testing Database of Information System using Conceptual Modeling

Bogdan Walek, Cyril Klimes

***Abstract*—**This paper focuses on testing database of existing information system. At the beginning we describe the basic problems of implemented databases, such as data redundancy, poor design of database logical structure or inappropriate data types in columns of database tables. These problems are often the result of incorrect understanding of the primary requirements for a database of an information system. Then we propose an algorithm to compare the conceptual model created from vague requirements for a database with a conceptual model reconstructed from implemented database. An algorithm also suggests steps leading to optimization of implemented database. The proposed algorithm is verified by an implemented prototype. The paper also describes a fuzzy system which works with the vague requirements for a database of an information system, procedure for creating conceptual from vague requirements and an algorithm for reconstructing a conceptual model from implemented database.

***Keywords*—**testing, database, relational database, information system, conceptual model, fuzzy, uncertain information, database testing, reconstruction, requirements, optimization

I. Introduction

TODAY, in the area of information systems development is very important to understand correctly customer's requirements, which are put on information system (further in text referred as IS), and afterwards to implement them correctly to ensure that created IS meets these requirements and brings the desired added value to the customer. This emphasis is also based on today's iterative approach to the design and development of the IS [1].

During the IS design and its subsequent implementation as well, there are a lot of changes and adjustments which can significantly affect the final quality of the developed IS. For this reason it is advisable to test the IS and its implementation against the original requirements for its creation.

An IS is normally connected to the database, so it is appropriate to focus on testing the IS database and its logical structure to the original requirements which are the results of analysis of the IS. This article is a continuation of articles [2] and [3].

II. Problem Formulation

As it was mentioned in the introduction, during the early analysis of the IS it is essential to understand the key customer's needs and requirements for the IS in development.

Bogdan Walek is with the Department of Informatics and Computers, University of Ostrava, 30. dubna 22, 701 03 Ostrava, Czech Republic (e-mail:bogdan.walek@osu.cz).

Cyril Klimes is with the Department of Informatics and Computers, University of Ostrava, 30. dubna 22, 701 03 Ostrava, Czech Republic (e-mail:cyril.klimes@osu.cz).

The requirements from the customer are very often formulated vaguely and in uncertain manner. This may lead to misunderstanding of the requirements or even to interpreting them in different meaning and this misunderstanding and misinterpretation is implemented into the final IS. For this reason it is advisable to check whether the final implementation and functionality of the IS are identical to the original customer's requirements for the IS. In some types of IS the database and its structure are two main parts to which the subsequent implementation is targeted [5], therefore it is appropriate to test the logical database structure against the IS requirements for its creation.

Database testing is particularly important because of possible data redundancy, which is caused by poor design of database logical structure and also by selecting inappropriate data types in columns of database tables, which can lead to an increase of size of the database. It is also important to check that the proper relationships between database tables are established as well, because well designed relationships are helping to maintain the consistency of data in the database [4].

The first part of this article shows one of the possible forms of representation of requirements, which can then be visualized using a conceptual model. Furthermore an algorithm will be proposed to compare the original requirements with the current implementation of the database. With the help of metadata (which are implemented in database) it is possible to reconstruct retroactively the conceptual model that contains entities, attributes, and relationships between entities (which correspond to the tables, columns and relationships implemented in the database), the output will be comparison of the two conceptual models. The conceptual model based on the requirements of database of the IS will be compared to the conceptual model of already implemented database. Another outcome will be a list of steps which are necessary to optimize the database. As the above description has shown, a relational database will be used and tested [4], [13].

III. Problem Solution

In some simple information systems a database is a main part of such IS [5], therefore customer's requirements for IS are in fact requirements for the IS database. A proposed solution will be introduced on this type of information systems, but it can be used in complex information systems environment as well.

First one of the possible forms of the representation of vaguely defined customer requirements will be introduced. This form will then be a help to design a logical database structure and also to display an appropriate visual form.

The proposed form of the representation of customer's requirements consists of these basic parts [3], [6]:

1. The list of entities, where for each entity a unique name and description (which defines the meaning and purpose of the entity) exist.
2. For each entity, a list of attributes and their vague description of type and size of data that are stored within the attribute is constructed.
3. Linking the entities together is defined using a special column that stores the link to linked entities.

From a definition of the proposed form of representation of requirements, an obvious similarity to the conceptual model that is suitable for visualization can be derived.

The conceptual model is a general model of a system that does not describe its implementation or its technological specifics, but defines the content of the future information system [7], [12].

Conceptual model works with these basic components [7], [12]:

1. Entity – it is an abstraction from the complexities of some domain. It represents some aspect or the object of the real world. Examples of entities: *student*, *subject*, *teacher*.
2. Relationship – identifies how two or more entities are related to one another. Relationships can be represented by verbs, linking two or more nouns (entities). Examples of relationships: *teaches* (between entities *teacher* and *subject*), *studies* (between entities *student* and *subject*).
3. Attribute – represents the properties of entities. Examples of attributes: *name*, *surname*, *age*, *email*, *phone* (attributes of entity *teacher*).

The proposed form of representation can be easily converted into a conceptual model, therefore, transfer to a conceptual model is used only for a better visualization of results and for better understanding of customer's requirements for future information system. Another reason for the appropriateness of the conceptual model is its ability to be relatively easily transformed into a logical data model and then into the physical model of the future database.

The following table shows part of the requirements for creating a database information system for the language school. The purpose of the IS is to record data about students and their attendance at the courses, teachers and their courses, and payment for the courses. The requirements are written in the proposed form [3]:

TABLE I
REPRESENTATION OF REQUIREMENTS OF THE DATABASE OF LANGUAGE SCHOOL INFORMATION SYSTEM

Name of entity	Description of entity	Name of attribute	Description of stored data
Student	Information about student	name	Short text
		surname	Short text
		age	Low number
		address	Short text
		phone	Low number
		course	Link to course
Course	Information about course	name	Short text
		location	Medium-length text
		start date	Date
		end date	Date
		description	Long text
		teacher	Link to teacher
Attendance	Information about courses attendance	date	Date
.		teaching plan	Long text
		course	Link to course
.		student	Link to student
Payment	Information about courses payment	amount	Low number
		date	Date
		payment method	Short text
		student	Link to student
Teacher	Information about teacher	name	Short text
		surname	Short text
		address	Short text
		phone	Low number
		login	Short text
		password	Short text

The table shows the different entities that were part of the requirements. Each entity contains a list of attributes, whose vague description of the data type and its length is shown in the last column of the table. Individual links to other entities are given together with attributes describing the entities and labeled with the name of the linked entities. The conceptual model containing entities and attributes in the table above is visually displayed in the following figure:

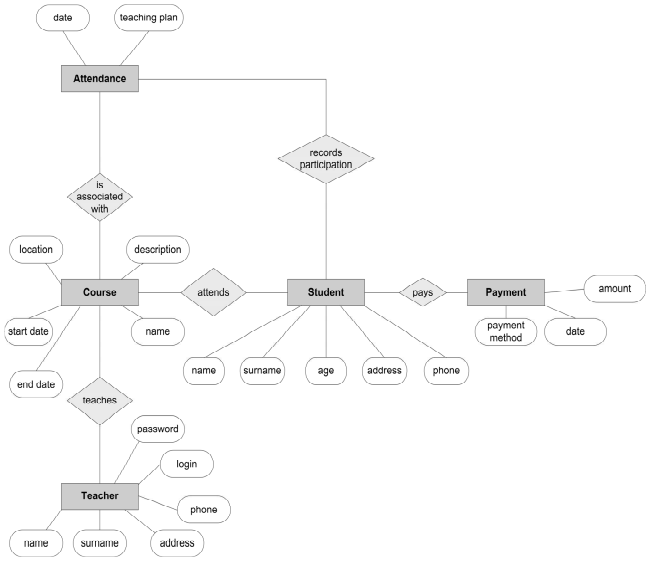

Fig. 1 Conceptual model of language school information system

Now a general model for supporting the decision process will be presented [8] [11]. This model is used in the algorithm for testing the database. The model is used to find suitable candidates for the data types of columns of database tables

based on vague descriptions in the proposed form of representation of the requirements and the expert knowledge [3].

The model consists of 4 main phases (M1, M2, M3, M4).

In – inputs into the process (uncertain information about attributes in entities).

Ou - the process outputs (a list of suitable candidates for data types of proposed columns in database tables).

M1 - The completion process and selection of relevant data. This process includes completion of user vague requirements with expert domain knowledge and selection of relevant data from the user's input requirements.

M2 - Creating a set of acceptable solutions based on the application of rules P and important information such as supplement the result of interference to input data or elimination of non-compliant data. The main goal of this phase is to create a set of all acceptable candidates for data types of the proposed columns in the database tables.

M3 - The process of modeling effects of admissible solutions. In this phase the relevance and the degree of suitability of candidates for data types of the proposed columns in the database tables are evaluated.

M4 - Selecting the most appropriate solution. In the last phase specific candidates for data types of the proposed columns in the database tables are selected.

This general model was already published and discussed by the scientific community [3] [6] [9].

IV. Database Testing Algorithm

Now the proposed algorithm for database testing will be described. The proposed algorithm goes through a table of requirements and is looking for individual entities and their attributes (along with the attributes that represent links to other entities). If a vague description of the data type attribute is found, then the algorithm uses fuzzy expert system for searches for suitable candidates for the data type and size of the attribute. The output of the algorithm is then compared to the actual data type found in the appropriate column of the database table from the implemented database.

Formally, the algorithm can be written as following [3] [6]:

```
Forall (entity_name in
db_requirements_table) Do
{
   Forall (attributes in
   entity(entity_name)) Do
   {

     If(is_relation_to_another_entity(
     attribute, entity) Then
     {
       write_req_relation(attribute,
       entity_name,entity_name2);
     }
     write_req_attribute(attribute,
     entity_name);

     candidates_data_types  =
     find_candidates_data_types(
     attribute);

     write_attribute_founded_candidates(
     attribute, candidates_data_types);
   }
}

Forall (db_table_name in database) Do
{
   Forall (attributes in
   db_table(db_table_name)) Do
   {
     If(is_foreign_key(attribute,
     db_table) Then
     {
       write_db_relation(attribute,
       db_table,db_table2);
     }
     write_db_attribute(attribute,
     db_table_name);

     atribute_type =
     get_attribute_type(attribute);

     write_db_attribute_type(
     attribute_type,attribute,
     db_table_name);
   }
}

Forall (attributes in db_attributes) Do
{
   compare_data_types(founded_candidates(
   attribute), attribute);
}

Forall (relations in db_relations) Do
{
   compare_relations(req_relations(
   attribute), db_relation);
}
```

At this stage there is a table of requirements on the database; this is visually displayed using a conceptual model. Also an expert system for the proposition of appropriate data types and algorithm for testing the implemented database is suggested. Now a procedure for reconstruction of the conceptual model from metadata of the implemented database will be proposed.

This procedure can be written down into several steps:

1. Connecting to the existing database and loading its metadata.
2. Browsing all database tables along with identification of individual columns, their names and data types.
3. Finding all relationships between database tables.
4. Transformation of all outputs into XML file with suitable form.
5. Visualization of the resulting XML file into a conceptual model.

The proposed algorithm shows that steps 2 and 3 are crucial for the final form of a conceptual model and any error in the implementation of these steps will have major impact on the resulting conceptual model.

The final step of testing the database is then comparison of the conceptual model created on the basis of requirements on database to the conceptual model reconstructed from metadata of the implemented database. One part of this step is also the creation of the list of necessary steps for possible optimization.

V. Results

Now, the realization of approach proposed for testing of the database on the grounds of already described example of the information system for the language school will be described. First the complete table containing the requirements for the database of this IS will be shown:

TABLE II
TABLE OF REQUIREMENTS FOR THE DATABASE OF LANGUAGE SCHOOL INFORMATION SYSTEM

Name of entity	Description of entity	Name of attribute	Description of stored data
Student	Information about student	name	Short text
		surname	Short text
		age	Low number
		address	Short text
		phone	Low number
		course	Link to course
Course	Information about course	name	Short text
		location	Medium-length text
		start date	Date
		end date	Date
		description	Long text
		teacher	Link to teacher
Attendance	Information about courses attendance	date	Date
.		teaching plan	Long text
		course	Link to course
.		student	Link to student
Payment	Information about courses payment	amount	Low number
		date	Date
		payment method	Short text
		student	Link to student
Teacher	Information about teacher	name	Short text
		surname	Short text
		address	Short text
		phone	Low number
		login	Short text
		password	Short text

The conceptual model based on this table of requirements will be displayed at the end of this chapter for comparison to the conceptual model implemented in database.

Now, the algorithm for finding the appropriate data types on the basis of vague description stored in table of requirements will be validated.

Expert system for realization of this step was implemented in the LFLC tool, which is described in [10].

TABLE III
CANDIDATES FOR DATA TYPES

Vague description of attribute	Proposed data type		
	VARCHAR	CHAR	TEXT
Short text	0.9	0.9	0.063
Medium-length text	0.26	0.26	0.157
Long text	0.043	0.043	0.93

In the table above we can see vague descriptions of attributes in the left column. On the right side of the table we can see proposed data types for each attribute. Evaluation of suitability degree of candidates for data types of attributes is represented by the value from the interval [0,1]. Suitability degree 0 means that proposed data type is completely unsuitable for the attribute. On the other side, suitability degree 1 means that proposed data type is perfectly suitable for the attribute. From the table we conclude that for attribute described as *short text* proposed data types VARCHAR and CHAR are very suitable, data type TEXT is unsuitable. For attribute described as *long text* proposed data types VARCHAR and CHAR are very unsuitable, but data type TEXT is very suitable.

In the following table the data types of the *Teacher* table columns (from the database of language school IS) are compared to found suitable candidates for the data types according to the vague attribute description from the table of demands:

TABLE IV
COMPARING SUITABLE CANDIDATES FOR DATA TYPES WITH ACTUAL DATA TYPES

Attribute of entity Student	Vague attribute description	Suitable candidates for data types	Actual column from Teacher table	Actual data type
name	Short text	VARCHAR, CHAR	name	VARCHAR
surname	Short text	VARCHAR, CHAR	surname	CHAR
address	Short text	VARCHAR, CHAR	address	TEXT

From this table it can be concluded that the data types of columns *name* and *surname* correspond to the candidates of the data types proposed by a decision making process, but column *address* does not correspond to the proposed candidates.

Now the steps for the reconstruction of a conceptual model from the implemented database will be described.

1. First it is necessary to connect to an existing database. This can be implemented in Java programing language using JDBC driver.
2. In this step it is necessary to find all database tables along with a description of the columns and data types. This can

be realized using *DatabaseMetaData getMetadata()* method. Process of collecting metadata from the implemented database is shown in the following figure:

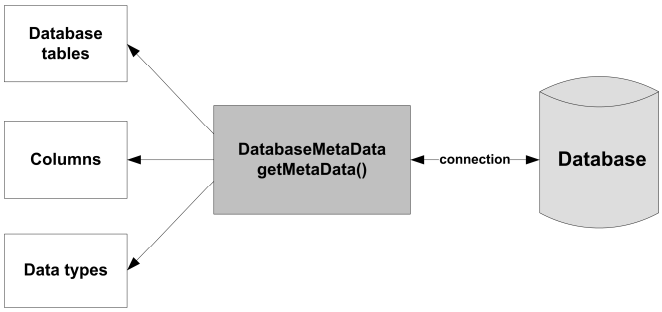

Fig. 2 Collecting metadata from the implemented database

3. It is also necessary to find and identify properly all relationships in database. This can be done using *getCrossReference()* method. We can call this method for all database tables that was identified in the previous step. As the output of this step we get collection of all relations in the implemented database. An example of finding relation between database tables *Student* and *Payment*:

```
ResultSet rs_cross =
meta.getCrossReference(null, null,
"Student", null, null, "Payment");

while (rs_cross.next())
{
    String pk_table =
    rs_cross.getString("PKTABLE_NAME");
    String pk_col =
    rs_cross.getString("PKCOLUMN_NAME");
    String fk_table =
    rs_cross.getString("FKTABLE_NAME");
    String fk_col =
    rs_cross.getString("FKCOLUMN_NAME");
}
```

The output is the relation, shown in the following figures:

```
Primary table: Student
Primary column: studentID
Foreign table: Payment
Foreign column: student_id
```

Fig. 3 Relation between tables Student and Payment

Fig. 4 Relation between tables Student and Payment in visual form

In the following table there is a logical structure of the actual database of the language school information system:

TABLE V
LOGICAL STRUCTURE OF THE ACTUAL DATABASE

Table name	Table purpose	Column name	Features
Student	Information about student	studentID	Primary key
		name	
		surname	
		age	
		street	
		city	
		zip	
		phone	
Course	Information about course	courseID	Primary key
		name	
		location	
		start_date	
		end_date	
		description	
		teacher_id	Foreign key (table Teacher)
Attendance		attendanceID	
		date	
		teaching plan	
		course_id	Foreign key (table Course)
		student_id	Foreign key (table Student)
Payment	Information about course payment	paymentID	Primary key
		amount	
		date	
		payment_method	
		course_id	Foreign key (table Course)
		student_id	Foreign key (table Student)
Teacher	Information about teacher	teacherID	Primary key
		name	
		surname	
		address	
		phone	
		login	
		password	

4. Next it is necessary to transform the acquired information into an appropriate XML file with suitable formalism. The resulting XML file contains metadata that are implemented in database. The content of the part of XML file looks as the following:

```
<?xml version=”1.0” encoding=”UTF-8” ?>
<table name=”Teacher” columns=”7”>
  <teacherID>1</teacherID>
  <name>John</name>
  <surname>Smith</surname>
  <address>Bolzanova 12, Praha
  11000</address>
  <phone>+420 111 222 333</phone>
  <login>john</login>
  <password>smith</password>
</table>
<table name=”Course” columns=”7”>
  <courseID>1</courseID>
```

```
  <name>English for beginners</name>
  <location>Praha</location>
  <start_date>1.1.2012</start_date>
  <end_date>31.12.2012</end_date>
  <description>This course is for
  beginners</description>
  <teacher_id>1</teacher_id>
</table>
<relationship>
  <primary_table>Teacher</primary_table>
  <primary_column>teacherID</primary_colum
  n>
   <foreign_table>Course</foreign_table>
  <foreign_column>teacher_id</foreign_colu
  mn>
</relationship>
```

5. The last step is to visualize the XML that can be implemented using appropriate methods in the Java programming language or using visualization tools.

The following figure shows a conceptual model created on the basis of vaguely defined requirements:

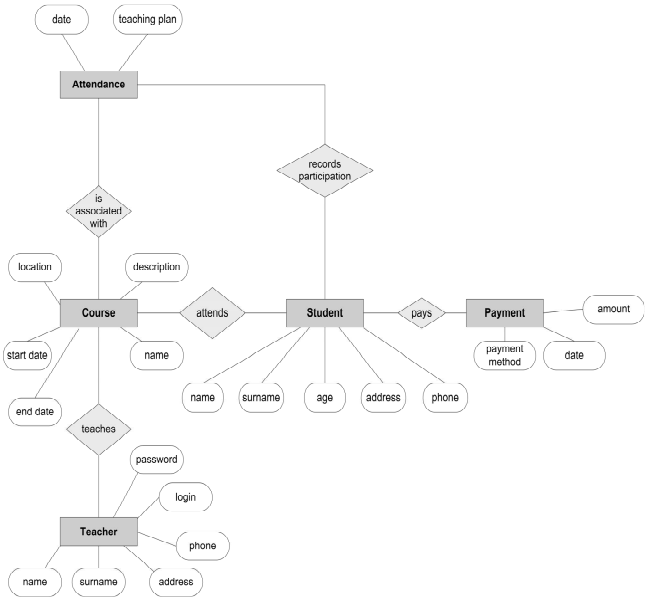

Fig. 5 Conceptual model based on a table of requirements

Next picture shows the conceptual model that is reconstructed from the implemented database, differences to previous conceptual model are marked, relationships between entities are generated from third step of the proposed procedure for the reconstruction of a conceptual model from the implemented database:

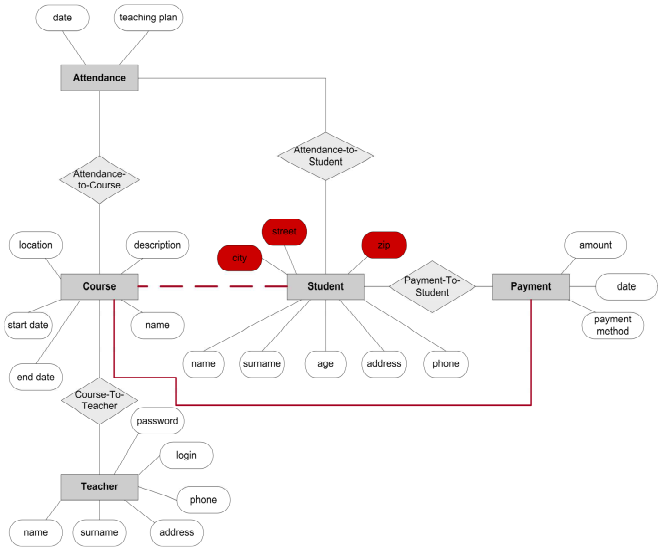

Fig. 6 Conceptual model reconstructed from the implemented database

These two conceptual models can be easily compared and the optimization proposals can be identified:

TABLE VI
OPTIMIZATION PROPOSALS FOR THE ACTUAL DATABASE

Table of demands	Existing database	Optimization proposal
	Useless relationship through secondary (foreign) key course_id of Payment table with primary key courseID of table Course	Remove this relationship and the foreign key course_id from table Payment
There is only one attribute address for storing information about adress of student in entity Student	Student table contains three attributes street, city, zip	Add attribute address to table Student Delete attributes street, city and zip from table Student
There is Link to entiy Course in entity Student	Missing relationship between tables Course and Student	Add foreign key course_id to table Student and add relationship to table Course

VI. CONCLUSION

In this paper, an algorithm to testing database using a conceptual model created on the basis of the requirements for the database of the IS and a conceptual model reconstructed from the database implemented in IS was designed. The part of the algorithm is a fuzzy expert system that allows proposition of the appropriate data types, attributes, based on the vague descriptions of the form of data stored in these attributes. Based on the outputs in the form of conceptual models it is possible to propose steps to optimize the database that is implemented. This procedure can be used both for testing of existing IS, and in teaching exercises in subject of relational database for easy verification of the correct logical structure of the database.

ACKNOWLEDGMENT

Presented topic is also a part of internal grant SGS10/PřF/2012, called Fuzzy modeling tools for analysis and design of information systems, at Department of Infomatics and Computers, University of Ostrava.

REFERENCES

[1] Rational Unified Process, http://www-01.ibm.com/software/awdtools/rup/, 2012.
[2] B. Walek, "Testování databáze informačního systému", Studentská vědecká konference 2011, Ostrava 2011, pp. 218-221.
[3] B. Walek, "Testing logical structure of the information system database", International Conference on Business Intelligence and Financial Engineering, Hong Kong 2011, to be published.
[4] L. Jan Harrington, *Relational database design and implementation*. Elsevier Inc., Burlington, 2009, ch. 4,5.
[5] S. Lauesen, *Software Requirements: Styles and Techniques*, Addison-Wesley Professional, Glasgow, 2002, ch. 1,2,3.
[6] J. Bartoš, B. Walek, P. Smolka, J. Procházka, C. Klimeš, "Fuzzy modeling tools for information system testing", 17th International Conference on Soft Computing Mendel 2011, Brno 2011, pp. 154-161.
[7] V. Řepa, *Analýza a návrh informačních systémů*, EKOPRESS, Praha 1999, pp. 146-159.
[8] C. Klimeš, "Model of adaptation under indeterminacy". In Kybernetika, Vol.47 (2011) No.3, Prague 2011, pp. 355-368.
[9] B. Walek, C. Klimeš, *Fuzzy tool for conceptual modeling under uncertainty*, Fourth International Conference on Machine Vision (ICMV 2011): Machine Vision, Image Processing, and Pattern Analysis, Proceedings of SPIE Vol. 8349, Bellingham 2012.
[10] H. Habiballa, V. Novák, A. Dvořák, V. Pavliska, "Using software package LFLC 2000", 2nd International Conference Aplimat 2003, Bratislava, 2003, pp. 355-358.
[11] J. Procházka, C. Klimeš, *Provozujte IT jinak: Agilní a štíhlý provoz, podpora a údržba informačních systémů a IT služeb*, Prague: Grada, 2011, ch. 8.
[12] A. Olivé, *Conceptual modeling on information systems*, Springer-Verlag Berlin Heidelberg, New York 2007, ch. 1.
[13] P. Atzeni, V. De Antonellis, *Relational Database Theory*, The Benjamin/Cummings Publishing Company, Inc., Redwood City, 1993, ch. 1.