Third International Conference
Modelling and Development of Intelligent Systems
October 10 - 12, 2013
“Lucian Blaga” University
Sibiu - Romania

A Method for Sampling Random Databases with Constraints

Letiţia Velcescu, Dana Simian, Marius Marin

Abstract

In this article, we propose a method for sampling the contents of random databases. This type of database is important either in modeling uncertainty or storing data whose values follow a probability distribution. Such uncertain or random data appear in a variety of scientific fields. In order to perform analysis or validate the properties of these databases it is useful to have samples of their instances. Our work introduces a formalization of relations in databases, which will provide a sound basis for the sampling algorithms that we will present. We classified the categories of atomic constraints and focused on them, thus obtaining algorithms that ensure that data satisfy each class of constraints defined in the database. For the modeling of the external constraints, our approach uses the graph theory. We illustrate the importance of the algorithm in the case of generating surface data.

1 Introduction

Generally, information is useful and valuable as it can provide support in decision making. Data behind the information may come from different sources and sometimes it is ambiguous or potentially erroneous. Even under these circumstances, databases have to provide the answers that at least converge to the real, expected information ([10]). The domain of databases that store this type of uncertain or error carrying information is closely related to the random databases field ([4], [9], [13]). The importance of this type of database is significant in a variety of research domains, like natural sciences, medicine, telecommunications and economics. It should also be noted in the domains of research that manage data provided by sensors.
When dealing with random databases, it is often necessary to obtain samples of them; such samples would allow further statistical analysis and the discovery of meaningful results on random data ([6]). A previous formal framework for database sampling in presented in [2]. Our work provides a method of sampling a database whose model might be complex. The relations in the database are considered with their corresponding attribute domains and with their specific constraints.
The database constraints are classified in distinct, atomic types according to their functionality. Each category of constraints is studied, leading to a data generation algorithm which ensures their satisfaction. The case of the external constraints is more complex and it was treated using elements of the graph theory ([11]). Overall, our method brings as benefit the possibility to obtain a database instance, given its structure, constraints and attribute domains.
In our work, we considered even the most particular types of associations that could appear in real database models, leading to constraints. As a consequence, the proposed algorithm supports even more complex database design cases, in which a foreign key is compound or the table is auto-referred.

In the second section of this paper we propose a formalization of the concepts of database theory which are relevant to our approach. The third part presents the elements of graph theory that are needed in resolving the external constraints; also, their application to our method is described. The fourth part presents the algorithm for the processing of each type of constraints category defined in the database's model. In the fifth section, we illustrate the importance of our algorithm in the case of generating surface data for the study of entropy transfer during sliding ([3]). The article ends with a conclusions section that briefly emphasizes the contributions of the presented research.

2 Relational Databases

In order to describe the method that we conceived for database sampling, we firstly provide a formalization which is close to the approach of our technique. In this respect, we start from the main concepts of database, relation, attributes and constraints.
As preliminaries from the database theory, we recall that a database is a set of relations. Each relation of the database is described by its corresponding relation schema ([1], [5]), which usually contains the attributes names, their domains and the primary key. We extend this concept, considering that a relation is described both by the schema (attributes and constraints) and the implementation (table).
Formally, consider a relational database, which is a set of relations, denoted by $DB = \left\{ R_i \middle| i \in \overline{1,n} \right\}$, where R_i are the relations and $n \in \mathbb{N}$ is the number of relations in the database. Each relation R_i has three associated sets $A(R_i)$, $C(R_i)$ and $T(R_i)$ representing the set of attributes, the set of constraints and, respectively, the relation's implementation or table.
The attributes of a relation R are represented by the set $A(R) = \left\{ A_i \middle| i \in \overline{1,n_A} \right\}$, where A_i are the attributes and $n_A \in \mathbb{N}$ is the number of attributes, denoting the relation's arity ([5]). Each attribute A_i has an associated domain of values $D(A_i)$.
As a remark, an attribute of a relation can be represented only by its associated domain.
The domain of values of the attribute A_i is the set $D(A_i) = \left\{ v_j \middle| j \in \overline{1,n_i} \right\}$, where v_j are values and $n_i \in \mathbb{N}$ is the number of values in the domain. We consider that each domain D_i contains the value *null* which denotes unknown information.
In order to classify the database's constraints in atomic types, it can be noticed that, actually, the not null constraint is a particular case of conditional (check) constraint and the primary key constraint is a combination of uniqueness and not null. Subsequently, the constraints in a database can be classified in three atomic types: conditional, unique and external. An external constraint refers to columns in different tables, so it represents the foreign key constraints.
Hence, we can consider that the constraints of a relation R are organized by categories into the tuple $C(R) = (C_C, C_U, C_E)$, where C_C, C_U and C_E are the sets of conditional, unique and respectively, external constraints.
In terms of the formalism that we are introducing for the design of our method, the three sets of constraints specified in a relation are defined below.
Definition 1. The conditional constraints of a relation R are represented by the set $C_C(R) = \left\{ P_{X_i} \middle| i \in \overline{1,n_C} \right\}$, where P_{X_i} are predicates associated to subsets X_i of attributes that belong to $A(R)$, $X_{i_1} \neq X_{i_2}$, $\left| X_{i_1} \right| \leq \left| X_{i_2} \right|$ for each i_1, $i_2 \in \overline{1,n_c}$, $i_1 < i_2$, and n_C is the number of conditional constraints, $n_C \in \mathbb{N}$, $n_C \leq 2^{|A(R)|}$.

In the definition above, we consider that the attributes of the set X_i are involved in the definition of the predicate P_{X_i}.

Further, we consider that a unique constraint is represented by the set of attributes on which the uniqueness is declared:

Definition 2. The unique constraints of a relation R are represented by the set $C_U(R)=\left\{X_i \middle| i\in\overline{1,n_U}\right\}$, where $X_i \in A(R)$ are subsets of attributes belonging to $A(R)$, $X_{i_1} \neq X_{i_2}$, $\left|X_{i_1}\right| \le \left|X_{i_2}\right|$ for each $i_1, i_2 \in \overline{1,n_U}$, $i_1 < i_2$, and n_U is the number of unique constraints, $n_U \in \mathbb{N}, n_U \le 2^{|A(R)|}$.

The external constraints in the third category involve a second relation R', two subsets of attributes belonging to $A(R)$, respectively $A(R')$ and a bijection between these subsets, like in the following:

Definition 3. The external constraints of a relation R are represented by the set $C_E(R)=\left\{\left(R', G_{X_i,X_i'}\right) \middle| i\in\overline{1,n_E}\right\}$, where R' are database relations, $X_i \in A(R)$ are subsets of attributes belonging to $A(R)$, $X_{i_1} \neq X_{i_2}$, $\left|X_{i_1}\right| \le \left|X_{i_2}\right|$ for each $i_1, i_2 \in \overline{1,n_E}$, $i_1 < i_2$, $X_i' \in C_U(R')$ are unique constraints that belong to R', $G_{X_i,X_i'}$ are the graphs of bijective functions $f_i : X_i \to X_i'$ and n_E is the number of external constraints, $n_E \in \mathbb{N}, n_E \le 2^{|A(R)|}$. For each pair $(A, A') \in G_{X_i,X_i'}$, the associated attribute domains are equal, i.e. $D(A) = D(A')$.

We emphasize that in all three definitions given before the indices of the subsets X_i are ordered ascending according to the number of elements in these subsets. This order is important in our sampling algorithm for optimization purposes.

In database theory, a relation schema can be implemented as a table. A relation schema can have multiple instances, which are the states of the relation at given moments in time ([5]). The table is the container of the current instance of the relation schema.

Formally, the table of a relation R is the set $T(R)=\left\{t_i \middle| i\in\overline{1,n_R}\right\}$, where t_i are tuples of values, $t_i \in \prod_{A_j\in A(R)} D(A_j)$ and n_R is the cardinality of the relation, $n_R \in \mathbb{N}$.

In order to describe the sampling method we conceived we need to use the projection operation from the relational algebra ([1]) and to define the extension of this operation.

The projection of a table $T(R)$ on a set X of attributes that belong to $A(R)$ is a function π_X that associates to each database's table a set of tuples:

$$\pi_X\left(T(R)\right)=\left\{t_i(X) \middle| i\in\overline{1,n_T}\right\}, \tag{1}$$

where $t_i(X)=\left(v_j \middle| A_j \in X\right)$ is the restriction of the tuple $t_i=\left(v_j \middle| j\in\overline{1,|A(R)|}\right)\in T(R)$ on the set of attributes X and n_T is the number of restricted tuples, $n_T \in \mathbb{N}, n_T \le |T(R)|$.

The table $T(R)$ corresponding to a relation R has to satisfy the constraints $C(R)$ of the same relation, i.e. the following implications hold:

$\forall P_X \in C_C(R),\ \forall t \in T(R) \Rightarrow P_X(t) = true$.

$\forall X \in C_U,\ \forall t_1 \neq t_2; t_1, t_2 \in T(R) \Rightarrow \pi_X(t_1) \neq \pi_X(t_2)$

$\forall \left(R', G_{X,X'}\right) \in C_E(R) \Rightarrow \pi_X\left(T(R)\right) \subseteq \pi_{X'}\left(T(R')\right)$

Further, we introduce the extension of this operation. We denote by $T_X = \left\{ T_i \middle| i \in \overline{1, n_{TX}} \right\}$ a table grouped by the projection on a set of attributes $X \subseteq A(R)$, where $T_i \subseteq T(R)$ are groups of tuples that partition the table $T(R)$:

$$\bigcup_{i \in \overline{1, n_{TX}}} T_i = T(R), \ \bigcap_{i \in \overline{1, n_{TX}}} T_i = \varnothing, \tag{2}$$

such that $\pi_X(t_1) = \pi_X(t_2)$ for each $t_1, t_2 \in T_i$, $\pi_X(t_1) \neq \pi_X(t_2)$ for each $t_1 \in T_{i_1}, t_2 \in T_{i_2}$, $i_1 \neq i_2$, and $n_{TX} \in \mathbb{N}$ is the number of groups.

Definition 4. Let be T(R) a table corresponding to a relation R and let $X_1, X_2 \in A(R)$ be sets of attributes. The extension of the projection of the table T(R) is the function:

$$\pi_{X_1}^{-1}\left(\pi_{X_2}\left(T(R)\right)\right) = \pi_{X_2}\left(T(R)\right) \times \prod_{A_i \in X_1 - X_2} D(A_i), \tag{3}$$

where X_1 is the set of attributes of the extension and X_2 is the set of attributes of the projection.

3 Graphs of Relations and Graphs of Attributes

3.1. Preliminaries

A multiset M_A is the graph of a function $m: A \to \mathbb{N}^*$ which assigns an order of multiplicity to each element of the support set A.

A directed graph is a pair $G = (V, M_E)$, where V are the vertices and M_E are the edges. The vertices of a graph G are represented by the set $V(G) = \left\{ v_i \mid i \in \overline{1, n_V} \right\}$, where $n_V \in \mathbb{N}$ is the number of vertices. The edges of the graph G are represented by the multiset $M_{E(G)}$, where $E(G) = \left\{ e_i = (v_1, v_2) \middle| i \in \overline{1, n_E} \right\}$ denotes a set that contains pairs of vertices, $v_1, v_2 \in V(G)$, and n_E is the number of these pairs ([7]).

The external neighbours of a vertex v of the graph G are represented by the set:

$$V_v^+ = \left\{ v' \in V(G) \middle| \exists e = (v, v') \in E \right\}. \tag{4}$$

Similarly, the internal neighbours of a vertex v are specified as:

$$V_v^- = \left\{ v' \in V(G) \middle| \exists e = (v', v) \in E \right\}. \tag{5}$$

The sets defined in formula (4) and (5) allow us to introduce the external and the internal degrees of a vertex v of the graph G. These degrees, denoted by $d_G^+(v)$, $d_G^-(v)$ specify the number of outbound, respectively inbound edges relative to the vertex v ([7], [11]). They are given by the following formulas:

$$d_G^+(v) = \sum_{v' \in V_v^+} m((v, v')), \tag{6}$$

$$d_G^-(v) = \sum_{v' \in V_v^-} m((v', v)). \tag{7}$$

We denote the set of vertices ordered descending by their internal degrees as the set $V_{d_G^-} = \left\{ v_i \middle| i \in \overline{1, |V|} \right\}$, where $d_G^-(v_{i_1}) \geq d_G^-(v_{i_2})$, $1 \leq i_1 < i_2 \leq |V|$.

The network of paths that connect two vertices $v, v' \in V$ is the set $N_{v,v'} = \left\{ P_i \middle| i \in \overline{1, n_P} \right\}$, where P_i are paths that begin at v and end at v' and $n_P \in \mathbb{N}$ is the number of possible paths.

A path that belongs to a network $N_{v,v'}$ is an ordered set $P=\left(e_i \middle| i\in\overline{1,l_P}\right)$, where $e_i \in E$ are edges of the graph, $e_i=\left(v_i,v_{i+1}\right)$, $v_1=v$, $v_{n+1}=v'$, and $l_P\in\mathbb{N}$ is the length of the path.
A network of closed paths is a network of paths $N_{v,v}$.

3.2. Graphs Modelling the Relational Databases

In our approach, we consider that a relational database DB has an associated pair of graphs $\left(G_A,G_R\right)$, where G_A the graph of is attributes and G_R is the graph of relations. We introduce these concepts in the definitions that follow in this paragraph.
Let DB be a database as defined in section 2, with $\left|DB\right|=n$, $n\in\mathbb{N}$ and $\left|A\left(R_i\right)\right|=m_i$ for each $R_i\in DB,\ i\in\overline{1,n}$.
Definition 5. The graph of attributes associated to a relational database is the graph G_A, where:

$$V\left(G_A\right)=\left\{\left(R,A\right)\middle|\forall R\in DB,\forall A\in A\left(R\right)\right\} \tag{8}$$

and

$$E\left(G_A\right)=\left\{\left(\left(R,A\right),\left(R',A'\right)\right)\middle|\forall R\in DB,\forall\left(R',G_{X,X'}\right)\in C_E\left(R\right),\left(A,A'\right)\in G_{X,X'}\right\}. \tag{9}$$

A vertex of the graph of attributes is specified by a pair composed of the relation name and an attribute of this relation. As we can notice, the support edges of G_A are defined by pairs of attributes that belong to external constraints.
Definition 6. The graph of relations correspondent to a relational database is the graph G_R, where $V\left(G_R\right)=\left\{R\middle|\forall R\in DB\right\}$ and

$$E\left(G_R\right)=\left\{\left(R,R'\right)\middle|\forall R\in DB,\forall\left(R',G_{X,X'}\right)\in C_E\left(R\right)\right\}. \tag{10}$$

The support edges of G_R are defined by pairs of relations that are involved in external constraints.
In both graphs defined above, if an edge appears more than once, then the multiplicity of that edge increases adequately.
In the following we consider that a database is correctly defined if the graph of attributes has only networks of open paths, meaning that

$$\forall v\in V\left(G_A\right)\Rightarrow N_{v,v}=\varnothing. \tag{11}$$

4 Sampling Algorithm

The algorithm we introduce refers to the sampling of data in a random database, which is defined as presented in section 2 of this paper. Briefly, the data in the generated tables need both to belong to the specified attributes domains and to satisfy the constraints defined on the relations in the database. More, we suppose that the database is correctly defined, in the sense given in (11). In order to process all categories of constraints, the algorithm involves two stages which will be presented as separate methods.
We define the algorithm ALG that constructs the tables corresponding to the relations of a random relational database as a pair of methods $ALG=\left(M_{CU},M_R\right)$, where M_{CU} represents the method that constructs the tables according to the conditional and unique constraints and M_R represents the method that manipulates the constructed tables according to the reference constraints.

4.1. Method to Sample Tables with Conditional and Unique Constraints

We describe the method M_{CU} using the following pseudo code representation.

Input: The database DB having the tables associated to each relation empty, i.e. $T(R)=\varnothing$, $\forall R \in DB$.

```
1.  BEGIN
2.  FOR $i \in \overline{1,|DB|}$
3.    $T = \pi_{\varnothing}(T(R_i))$;
4.    FOR $j \in \overline{1,|C_c(R_i)|}$
5.        $P_{X_j} \in C_c(R_i)$;
6.        $T = \pi_{X_j}^{-1}(T)$;
7.        $T = T - \{t \in T \mid \neg P_{X_j}(t)\}$;
8.    END FOR
9.    $T = \pi_{A(R_i)}^{-1}(T)$;
10.   FOR $j \in \overline{1,|C_u(R_i)|}$
11.       $X_j \in C_u(R_i)$;
12.       $T_{X_j} = T$;
13.       FOR $k \in \overline{1,|T_{X_j}|}$
14.           $T_k \in T_{X_j}$;
15.           $T_k = select(T_k)$;
16.       END FOR
17.       $T = \bigcup_{k=\overline{1,|T_{X_j}|}} T_k$;
18.   END FOR
19.   $T(R_i) = T$;
20. END FOR
END
```

Output: The database DB with the tables associated to each relation having data according to the defined conditional and unique constraints.

For each relation R_i of the database, the method above considers an auxiliary table T, initially with no rows and no columns (line 3). As the method uses the extension operation of projected tables, this table is initialized using the projection on the empty set of attributes.

For each conditional constraint defined on R_i, the method takes the attributes set X_j corresponding to the constraint's predicate P_{X_j} and applies the extension with X_j of the projection on the table T (line 6). The records that do not satisfy the predicate P_{X_j} are removed from the table T (line 7). After processing all the conditional constraints, the columns that were not included in these constraints are populated by the extension of the table T, which is actually the result of a projection (line 9).

Then, for each unique constraint defined on R_i, the method takes the corresponding attributes set X_j and considers the table T_{X_j} as the table T grouped by the projection on the attributes in X_j (line 12). For each group T_k in T_{X_j}, the function *select(T_k)* chooses a single random record (line 15), so that the unique constraint is satisfied. The selection method can use the simulation of any probability distribution (uniform, exponential, normal etc.), thus giving the database a random feature ([8], [12]).

The table T becomes the union of the atomic groups T_k (line 17). After processing all the unique constraints, the table T is associated to the relation R_i (line 19).

4.2. Method to Sample Tables with External Constraints

The method M_R in the algorithm ALG can be described using the following pseudocode representation.

Input: The database DB, with $|DB| = n$, having the tables associated to each relation generated with the method M_{CU}.

```
1.  BEGIN
2.   $used[\overline{1,n}] = 0;$
3.  FOR $R_i \in V_{d^-_{G_R}}$
4.    IF $used[i] = 0$
5.      $X = \{R_i\};$
6.    WHILE $X \neq \varnothing$
7.        $X_{next} = \varnothing;$
8.        $X_{d^-_{G_R}} = X;$
9.        FOR $R_{i_1} \in X_{d^-_{G_R}}$
10.           $T_{del} = \varnothing;$
11.           FOR $j \in \overline{1, C_E(R_{i_1})}$
12.               $(R_{i_2}, G_{X_j, X_j'}) \in C_E(R_{i_1});$
13.               FOR $k \in \overline{1, |T(R_{i_1})|}$
14.                   $t_k \in T(R_{i_1});$
15.                   IF $\pi_{X_j}(t_k) \not\subset \pi_{X_j'}(T(R_{i_2}))$
16.                       $T_{del} = T_{del} \cup \{t_k\};$
17.                   END IF
18.               END FOR
19.           END FOR
20.           IF $T_{del} \neq \varnothing$
21.               $T(R_{i_1}) = T(R_{i_1}) - T_{del};$
22.               $X_{next} = X_{next} \cup V^-_{R_{i_1}};$
23.           END IF
24.           $used[i_1] = 1;$
25.       END FOR
26.       $X = X_{next};$
27.   END WHILE
28.   END IF
29. END FOR
30. END
```

Output: The database DB with the tables associated to each relation having data according to the defined external constraints.

The method above marks a relation as used if it was treated by the algorithm, so that it will not be processed again. Each main iteration ensures that a connected component of the relations graph has been entirely treated. The relations in the database will be processed in descending order by the number of external constraints that refer to the relation, so that the relations referred most are treated first.
For each unused relation R_i in $V_{d_{\bar{G}_R}}$, an auxiliary set X is initialized with the element R_i (line 5). Again, we consider the relations set X ordered by the internal degree $X_{d_{\bar{G}_R}}$. For each relation R_{i_1} in this set, the algorithm takes all the relations R_{i_2} (line 9) that are referred by R_{i_1}; for each tuple t_k in the associated table $T\left(R_{i_1}\right)$, if the projection on the set of attributes X_j defining the constraint is not found in the projection of the table $T\left(R_{i_2}\right)$ on the corresponding attributes set $X_j{}'$, then the tuple t_k is included in the set T_{del} for deletion (line 16).
After processing all the constraints, the tuples in T_{del} are eliminated from $T\left(R_{i_1}\right)$ and the set X will contain the relations that depend through an external constraint on R_{i_1}. If no tuples have been eliminated, then the relations that depend on R_{i_1} will not be influenced by the current relation, so they are not included in the set X for further processing.
The algorithm can be used to generate random data for given models for the purpose of testing out various theories and their results.

5 Case Study: Generating Surface Data for the Study of Entropy Transfer during Sliding

In this section, we present a concrete example in which the algorithm described before can be used. The case study taken into account needs only one method from the algorithm ALG but it allows us to understand easier how the algorithm works. Many more complex examples using the proposed algorithm will be given in a future article.
In the study of entropy transfer during sliding presented in [3], the experiments and their subsequent computations started out with a generated surface. It was considered that this surface contained sites in one dimension and each site was described by its height. The data structure capable to retain the information of the surface is a simple one dimensional array which maps the heights as values and their corresponding sites as indices. Therefore, for a number of $n \in \mathbb{N}^*$ sites we need an array of length n which will contain at each i-th element the height $h_i \in \mathbb{R}$ of the i-th site.
We expressed the definition of the surface using an informatics data structure, so we consider transforming it further so it can match the relational database theory. We need a simple relation R in order to map the contents of the array into its associated table. The attributes set of R would minimally be $A(R) = (A_1, A_2)$, where A_1 represents the site and A_2 is the corresponding height. The relation needs to have a unique constraint defined on the attribute A_1, in order to ensure that the same site will not have two different heights. The constraints need to be $C = (C_C, C_U, C_E)$, where $C_U = \{\{A_1\}\}$ is the set of unique constraints. We will consider the external constraints to be unnecessary here, as in this case the data model is very simple. Therefore $C_E = \varnothing$. The conditional constraints can be more or less restrictive, depending on the purpose of the required data, so we will only consider that the height at each site needs to be strictly positive, $h_i > 0$,

$i \in \overline{1,n}$. Thus, the set of conditional constraints is formally defined as $C_C = \left\{P_{\{A_2\}}\right\}$, where $P_{\{A_2\}} = \left\{A_2 > 0\right\}$.

We have a relation schema, so next we will need actual values. We consider the domains associated to the attributes of R to be $D(A_1) = \left\{i \in \mathbb{N}^* \,|\, i \le 1000\right\}$ and $D(A_2) = [-2;2] \subset \mathbb{R}$.

There are two remarks that can instantly be made so far, but which will not prevent by any means the algorithm to run its course without error. The first observation is that $D(A_2)$ is an infinite set in the theory of real numbers so we cannot determine all the numbers that belong to it. In informatics, however, a number can only be expressed with a certain precision of digits, which is also limited by the memory available to store it, so in this case infinite numbers can only be approximated more or less.

Therefore, our domain will contain only the numbers in the given interval which have, for example, a precision of two digits, thus making it a finite set of numbers. The second observation is that the conditional constraint $A_2 > 0$ can be eliminated if we consider the domain $D(A_2)$ to contain strictly positive real numbers: $D(A_2) = (0;2]$. The reason for which the constraint is defined is to illustrate the possibility of introducing conditional constraints.

The algorithm performs the cross product between the domains of the attributes A_1 and A_2 and keeps only the tuples that satisfy the conditional constraints. In our case, the pairs that contain negative heights are discarded and the result looks like:

$$1\ \begin{matrix}0.01\\0.02\\\vdots\\2.00\end{matrix}\quad 2\ \begin{matrix}0.01\\0.02\\\vdots\\2.00\end{matrix}\quad \dots\quad 1000\ \begin{matrix}0.01\\0.02\\\vdots\\2.00\end{matrix}\,. \tag{11}$$

The valid pairs are grouped by the projection on the set of attributes $X = \left\{A_1\right\}$, which represents the single unique constraint that we have defined. The algorithm selects a single pair from each group through the random selection method provided at the corresponding step of the algorithm. In our case we consider that the method makes a uniform selection, meaning that the pairs from each group have equal chances of being selected. The result could resemble: $\left\{(1,0.93),(2,1.07),\dots,(1000,1.40)\right\}$ or $\left\{(1,0.03),(2,0.11),\dots,(1000,1.97)\right\}$ or another of the 200^{1000} possible combinations.

6 Conclusion

The algorithm presented in this paper allows obtaining samples of databases with complex design that might also have to store uncertain or random data. Even for attributes domains having small cardinality, a large database can be generated. The random feature of our algorithm is justified by two aspects: the values of at least one domain can follow a certain probability distribution and the selection of a unique row can be randomly performed using a specific algorithm. In our future work, we will provide an optimization of the processing of the very restrictive constraints, which could result in the generation of empty tables due to the random selection of tuples.

The main contribution of the research presented in this article consists in the fact that, using our approach on relations and constraints, it provides the suitable means to sample relational databases. The benefit of such a sample is that it can be used for various purposes, including analysis and testing. As an example, we considered the case study regarding the generation of surface data.

The algorithm completely processes the databases constraints, allowing the implementation of any model, as it exhaustively covers all the foreseeable situations in database modeling.

Acknowledgements

This work was supported by the strategic grant POSDRU/89/1.5/S/58852, Project "Post-doctoral programme for training scientific researchers" co financed by the European Social Found within the Sectorial Operational Program Human Resources Development 2007-2013.

References

1. Abiteboul, S.; Hull, R.; Vianu, V., *Foundations of Databases*; Addison-Wesley, 1995.
2. Bisbal, J.; Grimson, J.; Bell, D. A Formal Framework for Database Sampling. *Information and Software Technology* **2005**, *47*, 819-828.
3. Fleurquin, P.; Fort, H.; Kornbluth, M.; Sandler, R.; Segall, M.; Zypman, F. Negentropy Generation and Fractality in the Dry Friction of Polished Surfaces. *Entropy* **2010**, *12*, 480-489.
4. Katona, G.O.H. Random Databases with Correlated Data. *Conceptual Modelling and Its Theoretical Foundations* **2012**, *7260*, 29–35.
5. Kifer, M.; Bernstein, A.; Lewis, P. *Database Systems. An Application-Oriented Approach*; Addison Wesley, 2005.
6. Lutu, P. Database Sampling for Data Mining. *Encyclopedia of Data Warehousing and Mining* **2009**, 604-609.
7. Popescu, D. *Combinatorics and Graph Theory* (in Romanian); Romanian Mathematical Society Publishing House, Bucharest, 2005.
8. Ross, S. *Simulation*; Academic Press, San Diego, London, 1997.
9. Seleznjev, O.; Thalheim, B. Random Databases with Approximate Record Matching. *Methodology and Computing in Applied Probabability* **2008**, *12*, 63-89.
10. Tchangani, A.P. A Model to Support Risk Management Decision-Making. *Studies in Informatics and Control* **2011**, *20*, 209-220.
11. Tomescu, I. *Combinatorics and Graph Theory* (in Romanian); University of Bucharest Publishing House, Bucharest, 1978.
12. Văduva, I. *Simulation Models* (in Romanian); University of Bucharest Publishing House, Bucharest, 2005.
13. Velcescu, L. Relational operators in heterogeneous random databases. *IEEE Proceedings of the 11th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, IEEE Computer Society Press, **2009**, 407-413.

VELCESCU LETIŢIA
University of Bucharest
Department of Informatics
14, Academiei, 010014, Bucharest
ROMANIA
E-mail: letitia@fmi.unibuc.ro

DANA SIMIAN
University "Lucian Blaga" of Sibiu
Department of Mathematics and Informatics
15, Ratiu, Sibiu
ROMANIA
E-mail: dana.simian@ulbsibiu.ro

MARIUS MARIN
ROMANIA
E-mail: marin_marius_89@yahoo.ro