МАТЕМАТИКА И МАТЕМАТИЧЕСКО ОБРАЗОВАНИЕ, 2007
MATHEMATICS AND EDUCATION IN MATHEMATICS, 2007
Proceedings of the Thirty Sixth Spring Conference of
the Union of Bulgarian Mathematicians
St. Konstantin & Elena resort, Varna, April 2–6, 2007

FINDING THE FRACTIONAL AND THE FRACTIONAL BRANCHING DEPENDENCIES IN DATABASES*

Tsvetanka L. Georgieva

In the present paper the fractional and the fractional branching dependencies are considered. Some properties of these dependencies are examined. An algorithm for finding all fractional dependencies between a given set of attributes and a given attribute in a database relation is proposed.

1. Introduction. Functional dependencies are relationships between attributes of a database relation. The discovery of the functional dependencies that reflect the present content of the relation is an important database analysis technique. The basic motivation for discovery of the functional dependencies, which hold in the current instance of a relation, is discovery of valuable knowledge of the structure of the relation instance.

The functional dependency requires the values in a given set of attributes to determine uniquely the value in a given attribute. In [1] a branching dependency that is a more general dependency than the functional dependency is introduced. The paper [2] contains investigations concerning this dependency. The branching dependency allows to determine the maximal number of the different values in a given attribute b corresponding to one or more different values in a given set of attributes A in the relation. We obtain fractional branching dependency by adding the requirement b functionally determined A. In case of the fractional dependency we can determine the maximal number of the different values in b, corresponding to p in number different values in the attributes of A, but we consider only these values in b, that remain after the elimination of the values in b which lead to the maximum for $p-1$. This knowledge is additional information that may be useful for analyzing the current content of the database.

In the present paper a fractional dependency and fractional branching dependency are defined and some properties of these dependencies are examined. An algorithm for finding all fractional dependencies between a given set of attributes and a given attribute is proposed. The rest of the paper is organized as follows. In section 2 we present a brief survey on the related work. In section 3 the task for discovering of all fractional dependencies is formulated. In section 4 an algorithm for finding of all

*The work was supported partially by the Bulgarian National Science Fund under Grant IO-03-02/2006

Key words: branching dependency; fractional dependency; functional dependency; database; knowledge discovery.

2000 Mathematics Subject Classification: 68P15.

fractional dependencies is described. Some results from the execution of the algorithm are represented.

2. Related work. The basic definitions and properties connected with the functional dependencies in relational databases are represented in detail in [3, 5]. Some theorems valid for functional dependencies are generalized to branching dependencies in [1]. Moreover, some implications among branching dependencies are investigated. In [2] the estimations for the minimal number of tuples in a relation that results the sets of attributes, which (p, q)-depend on sets A of attributes are found. In [4] the task for finding all branching dependencies between a given set A of attributes and a given attribute b is considered. For that purpose, a minimum branching dependency is defined in a manner that the validity of all branching dependencies between A and b can be established if all minimal branching dependencies are known. Moreover, some properties of the branching dependencies are examined that allow to prune some values of p and q during the search of the branching dependencies between A and b and to create an efficient algorithm.

In the present paper the task for finding all fractional dependencies between a given set A of attributes and a given attribute b is considered.

3. Fractional and fractional branching dependencies. Let R be a database relation and let Ω be the set of attributes of the relation R. The number of the attributes is $|\Omega| = n$. We say that the *functional dependency* (*FD*) $A \to B$ *holds* or *is valid* in R, if for any two tuples r, $s \in R$ we have: if $r(a_k) = s(a_k)$ for all $a_k \in A$, $A = \{a_1, \ldots, a_l\}$, $A \subset \Omega$, $k = 1, \ldots, l$, then $r(b) = s(b)$ за $\forall b \in B$, $B \subset \Omega$. The branching dependency is defined in [1] by the following way: we say that b (p, q)-*depends* on A if there are no $q+1$ tuples such that they contain at most p different values in each $a_k \in A$, $k = 1, \ldots, l$, but $q+1$ different values in b. We also say that (p, q)-branching dependency holds and write $A \xrightarrow{(p,q)} b$.

Let π be the projection operator, δ be the duplicate-elimination operator and let $A \subset \Omega$, $b \in \Omega$, $b \notin A$ and let $1 \leq p \leq q$ be integers. We consider that $A \xrightarrow{(p,|\delta(\pi_b(R))|)} b$ is valid, if each $|\delta(\pi_b(R))|$ in number tuples with $|\delta(\pi_b(R))|$ different values in b have at least p different values in the attributes of A. In [4] we defined a minimal branching dependency: we say that the branching dependency $A \xrightarrow{(p,q)} b$ for $1 \leq p \leq q$ is *minimal*, if for $q < |\delta(\pi_b(R))|$ the dependency $A \xrightarrow{(p,q_1)} b$ does not hold for each $q_1 < q$, $p \leq |\delta(\pi_{a_1,\ldots,a_i}(R))|$ and for $q = |\delta(\pi_b(R))|$ the dependency $A \xrightarrow{(p_1,q)} b$ does not hold for each $p_1 < p$, where p, q, p_1, q_1 are integers.

Let σ be the selection operator, $\bowtie$ be the natural join operator, $v_i = (v_{i1}, \ldots, v_{il})$ be any element of the set $\delta(\pi_{a_1,\ldots,a_l}(R))$ and $d_A = |\delta(\pi_{a_1,\ldots,a_l}(R))|$, $d_b = |\delta(\pi_b(R))|$. We suppose that FD $A \to b$ is not valid. We consider the following example:

Example 1. There are given d_A in number tasks. Each task is subdivided to several subtasks. Each subtask can be used to conclude one or more tasks. The total number of the subtasks is d_b. The complexity of one task is determined by the number of its subtasks. For each $1 \leq p \leq d_A$ we have to find the minimal number subtasks that we need to conclude that there are p in number tasks with the maximal *total* number of subtasks (i.e. with maximal complexity).

Finding the minimal branching dependencies can solve this problem.

We consider the following variant of the example 1: Each subtask can be used to solve *exactly one* task.

This case corresponds to special type branching dependencies. We define these dependencies by the following way.

Definition 1. *A branching dependency* $A \xrightarrow{(p,q)_{fb}} b$ *is called* fractional branching dependency*, if it is a minimal branching dependency and the functional dependency* $b \to A$ *holds.*

Proposition 1. *We suppose that the functional dependency* $b \to A$ *holds and the integers* $c_1, c_2, \ldots, c_{d_A}$*, where*

$$c_k = \max\{|\delta(\pi_b(\sigma_{a_1=v_{i1}\,and\ldots and\,a_l=v_{il}}(R_k)))|for\ i = 1, \ldots, d_A\}, \quad k = 1, \ldots, d_A,$$

are sorted in descending order, i.e. $1 \le c_{k+1} \le c_k$*, for each* $k = 1, \ldots, d_A - 1$*. If the value* c_k *is obtained for the values* $(v_{k1}, \ldots, v_{kl})$ *of the attributes of* A*, then the relation* R_{k+1} *is obtained by the following way:* $R_1 = R$*,* $R_{k+1} = R_k \backslash (R_k \triangleright\!\triangleleft \pi_b(\sigma_{a_1=v_{k1}\,and\ldots and\,a_l=v_{kl}}(R_k)))$ *for* $k = 1, \ldots, d_A$*. Then the set*

$$FB = \left\{ A \xrightarrow{(1,c_1)_{fb}} b, A \xrightarrow{(2,c_1+c_2)_{fb}} b, \ldots, A \xrightarrow{(p,c_1+c_2+\ldots+c_p)_{fb}} b, \right.$$
$$\left. where \quad 1 \le p \le d_A; \quad c_1 + c_2 + \cdots + c_p = d_b \right\}$$

contains all fractional branching dependencies between A *and* b*.*

Proof. We apply induction on p. For $p = 1$ we obtain a valid fractional branching dependency, since c_1 is the maximal number of the different values in the attribute b, corresponding to the values in the attributes of A. We suppose that for $p = k$ BD $A \xrightarrow{(k,c_1+c_2+\ldots+c_k)_{fb}} b$, obtained by the following way, is valid fractional branching dependency. We have to proof that for $p = k + 1$, BD $A \xrightarrow{(k+1,c_1+c_2+\ldots+c_{k+1})_{fb}} b$ is also valid fractional branching dependency. For this purpose, first we assume that it is not valid, i.e. there exist $c_1 + c_2 + \cdots + c_k + c_{k+1} + 1$ tuples with at most $k + 1$ different values in the attributes of A and $c_1 + c_2 + \ldots + c_k + c_{k+1} + 1$ different values in the attribute b. We denote these tuples by r_1, r_2, …, r_c,r_{c+1}, …, $r_{c+c_{k+1}}$, $r_{c+c_{k+1}+1}$, where $c = c_1 + c_2 + \cdots + c_k$. The tuples $r_1, r_2, \ldots, r_c, r_{c+1}$ are $c + 1$ in number, consequently they have at least $k+1$ different values in the attributes of A, since otherwise we obtain contradiction with the induction assumption. Then, r_j, $j \in \{c + 2, \ldots, c + c_{k+1} + 1\}$ coincides in the attributes of A with some of the tuples r_i, $i \in \{1, \ldots, c+1\}$, consequently we obtain contradiction with the selection of some of the values $c_1, c_2, \ldots, c_k, c_{k+1}$.

On the other hand, if $A \xrightarrow{(k,q)_{fb}} b \notin FB$, then it is not valid, because if we suppose that $q \ne c_1 + c_2 + \cdots + c_k$, then the considered dependency is either invalid BD, or it is not minimal BD. Hence, it is not valid fractional branching dependency. □

Consequently, the values c_k, $k = 1, \ldots, d_A$ can be obtained with SQL (*Structured Query Language*) query from the following type:

```
SELECT COUNT(DISTINCT b) AS ck
FROM R
GROUP BY a1, ..., al
ORDER BY ck DESC
```

We consider other variant of the example 1: Each subtask can be used to solve one or more tasks. For each $1 \leq p \leq d_A$ we have to find the minimal number of subtasks that we need to solve p in number tasks. Each p-th sequent task is selected such that for its conclusion to remain the maximal number *unsolved* subtasks.

Finding the branching dependencies does not lead to desision exemplary problems of this type that motivates the definition of other type dependencies.

Definition 2. *We suppose that $A \subset \Omega$, $b \in \Omega$, $b \notin A$ and $1 \leq p < q_p$ are integers and that the following conditions hold:*

1. *for $p = 1$, $q_1 < d_b$: there are no $q_1 + 1$ tuples with equal values in the attributes of A and $q_1 + 1$ different values in the attribute b and q_1 is the minimal with this property;*
2. *for $p = 2$, $q_2 < d_b$: each $q_2 + 1$ tuples with $q_2 + 1$ different values in b, that contain q_1-element subset with equal values in A and q_1 different values in b, they have at least 3 different values in A and q_2 is the minimal with this property;*
3. *for $2 < p \leq d_A$, $q_p < d_b$: each $q_p + 1$ tuples with $q_p + 1$ different values in b, that contain q_{p-1}-element subset with exactly $p - 1$ different values in A and q_{p-1} different values in b, they have at least $p + 1$ different values in A and q_p is the minimal with this property;*
4. *for $q_p = d_b$: each q_p tuples with q_p different values in b, that contain q_{p-1}-element subset with exactly $p - 1$ different values in A and q_{p-1} different values in b, they have at least p different values in A.*

Then we say that (p, q_p) fractional dependency between A and b holds and write $A \xrightarrow{(p,q_p)_f} b$.

The central task we consider is with a given set A of attributes and an attribute b, to find all fractional dependencies between A and b.

We suppose that $c_1 = \max\{|\delta(\pi_b(\sigma_{a_1=v_{i1}\,and\ldots and\,a_l=v_{il}}(R)))|$ for $i = 1, \ldots, d_A\}$, $V^{(1)}$ contains these values in the attributes of A, for which the value c_1 is obtained, i.e.

$$V^{(1)} = \{v_j = (v_{j1}, \ldots, v_{jl}),\ \text{where } (v_{j1}, \ldots, v_{jl}) \in \delta(\pi_{a_1,\ldots,a_l}(R)) \text{ and } |\delta(\pi_b(\sigma_{a_1=v_{j1}\,and\ldots and\,a_l=v_{jl}}(R)))| = c_1\}.$$

We consider the tree T with the nodes obtained by the following way (Fig. 1):

- the node (*all*, d_b) is the root of the tree T;
- (v_j, c_1) for $v_j \in V^{(1)}$ are the nodes of the tree T from the first level, i.e. the child nodes of the root (*all*, d_b);
- For each element $v_j \in V^{(1)}$ we obtain the relation
$$R_2^j = R \backslash (R \triangleright\!\triangleleft \pi_b(\sigma_{a_1=v_{j1}and\ldots and\,a_l=v_{jl}}(R))).$$
We find the value of $c_2^j = \max\{|\delta(\pi_b(\sigma_{a_1=v_{i1}\,and\ldots and\,a_l=v_{il}}(R_2^j)))|$ for $i = 1, \ldots, d_A\}$. We form the set $V_j^{(2)} = \{v_i = (v_{i1}, \ldots, v_{il})$, where $(v_{i1}, \ldots, v_{il}) \in \delta(\pi_{a_1,\ldots,a_l}(R_2^j))$ and $|\delta(\pi_b(\sigma_{a_1=v_{i1}\,and\ldots and\,a_l=v_{il}}(R_2^j)))| = c_2^j\}$. Then, (v_i, c_2^j), $v_i \in V_j^{(2)}$ are the child nodes of the node (v_j, c_1).
- If (v_j, c_k) is a node on level k, $k \geq 2$, then its child nodes are obtained by using the relation $R_{k+1}^j = R_k^j \backslash (R_k^j \triangleright\!\triangleleft \pi_b(\sigma_{a_1=v_{j1}and\ldots and\,a_l=v_{jl}}(R_k^j)))$, by computing the value of $c_{k+1}^j = \max\{|\delta(\pi_b(\sigma_{a_1=v_{i1}\,and\ldots and\,a_l=v_{il}}(R_{k+1}^j)))|$ for $i = 1, \ldots, d_A\}$, by forming

the set $V_j^{(k+1)} = \{v_i = (v_{i1}, \ldots, v_{il})$, where $(v_{i1}, \ldots, v_{il}) \in \delta(\pi_{a_1,\ldots,a_l}(R_{k+1}^j))$ and $|\delta(\pi_b(\sigma_{a_1=v_{i1}\, and \ldots and\, a_l=v_{il}}(R_{k+1}^j)))| = c_{k+1}^j\}$. Then (v_i, c_{k+1}^j), $v_i \in V_j^{(k+1)}$ are the child nodes of the node (v_j, c_k).

Since on each step some tuples of the relation are deleted, for each j there exists k_j, for which $R_{k_j}^j = \emptyset$, from where the leaves of the tree T are obtained.

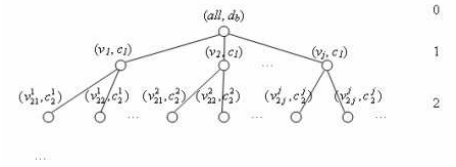

Fig. 1. Common view of the tree T, obtained by the described way

We suppose that $c_2 = \max\limits_j\{c_2^j$ for $\forall v_i \in V_j^{(2)}, (v_i, c_2^j)$ that is a node of T on the second level$\}$, then we consider the set of the nodes on the second level that contain this value c_2 and we find $V_2 = \{v_i$ for $\forall (v_i, c_2), v_i \in V_j^{(2)}\}$. We obtain the value $c_3 = \max\limits_j\{c_3^j$ for $\forall (v_i, c_3^j)$ that is a child node of $(v_j, c_2), v_j \in V_2\}$, we find $V_3 = \{v_i$ for $\forall (v_i, c_3), v_i \in V_j^{(3)}$ that is a child node of $(v_j, c_2), v_j \in V_2\}$ and so on, $c_p = \max\limits_j\{c_p^j$ for $\forall (v_i, c_p^j)$ that is a child node of $(v_j, c_{p-1}), v_j \in V_{p-1}\}$; $V_p = \{v_i$ for $\forall (v_i, c_p), v_i \in V_j^{(p)}$ that is a child node of $(v_j, c_{p-1}), v_j \in V_{p-1}\}$.

Proposition 2. *The set* $B = \left\{ A \xrightarrow{(1,c_1)_f} b,\ A \xrightarrow{(2,c_1+c_2)_f} b,\ \ldots,\ A \xrightarrow{(p,c_1+c_2+\ldots+c_p)_f} b,\ \text{where } 1 \le p \le d_A;\ c_1+c_2+\cdots+c_p = d_b \right\}$ *contains all fractional dependencies between* A *and* b.

Proof. We apply induction on p. For $p = 1$ we obtain a valid fractional dependency, because c_1 is the maximal number of the different values in b, corresponding to the values in the attributes of A. We assume that for $p = k$ the dependency obtained by the described way $A \xrightarrow{(k,c_1+c_2+\ldots+c_k)_f} b$ is valid fractional dependency. We suppose that $r_1, r_2, \ldots, r_c, r_{c+1}$ are $c+1$ in number tuples with $c+1$ different values in b, which have at least $k+1$ different values in A, where $c = c_1 + c_2 + \cdots + c_k$ and they contain $(c - c_k)$-element subset with exactly $k-1$ different values in A and $c - c_k$ different values in b. We have to proof that for $p = k+1$, $A \xrightarrow{(k+1,c_1+c_2+\ldots+c_{k+1})_f} b$ is also valid fractional dependency. Therefore, we consider the tuples $r_1, r_2, \ldots, r_c, r_{c+1}, \ldots, r_{c+c_{k+1}}, r_{c+c_{k+1}+1}$ with $c + c_{k+1} + 1$ different values in b. If we assume that these tuples have at most $k+1$ different values in A, then we obtain contradiction with the selection of some of the values $c_1, c_2, \ldots, c_k, c_{k+1}$.

In the next step of the proof we assume that there exists a valid fractional dependency $A \xrightarrow{(k,q)_f} b \notin B$. If $k \in \{1, \ldots, p\}$, then there exists a valid fractional dependency

$A \xrightarrow{(k,c_1+c_2+\ldots+c_k)_f} b \in B$, hence, $A \xrightarrow{(k,q)_f} b$ is not valid. If the fractional dependency $A \xrightarrow{(p,d_b)_f} b \in B$ is valid, then for $p < k \leq d_A$ the dependency $A \xrightarrow{(k,q)_f} b$ is not valid. □

4. An algorithm for finding all fractional dependencies. In the algorithm represented as Algorithm 1 we use the function *Get_ BD_ tree*(R). This function creates a relation *R_ tree*(Id, a_1, …, a_l, c, *lvl*, *Parent*), that contains the described tree structure: Id is unique identifier for the tuples in the relation *R_ tree*; c stores the values c_1, c_2, …, c_p; *lvl* determines the level of the relevant node of the tree structure; *Parent* stores NULL for the root of the tree and the value of the attribute Id of the parent of the relevant node for the other nodes of the tree.

Algorithm 1
Input: relation R with a set of attributes Ω; chosen set of attributes $A = \{a_1, \ldots, a_l\}$; an attribute b, $b \notin A$
Output: all fractional dependencies between A and b

1. $R\_tree = Get\_BD\_tree(R)$
2. $c_1 = \pi_c(\sigma_{lvl=1}(R\_tree))$
3. **Output** $A \xrightarrow{(1,c_1)_f} b$
4. $p = 2$
5. $c_p = \max\{\pi_c(\sigma_{lvl=p}(R\_tree))\}$
6. $q = c_1 + c_p$
7. **Output** $A \xrightarrow{(p,q)_f} b$
8. $Vtree_p = \pi_{Id}(\sigma_{lvl=p \text{ and } c=c_p}(R\_tree))$
9. **While** $Vtree_p \neq \emptyset$
10. $p = p + 1$
11. $c_p = \max\{\pi_c(\sigma_{lvl=p \text{ and } Parent \in Vtree_{p-1}}(R\_tree))\}$
12. $q = q + c_p$
13. **Output** $A \xrightarrow{(p,q)_f} b$
14. $Vtree_p = \pi_{Id}(\sigma_{lvl=p \text{ and } c=c_p \text{ and } Parent \in Vtree_{p-1}}(R\_tree))$

The temporary relation $V(Id, a_1, \ldots, a_l, c, lvl, Parent)$ is used in the following function. The attribute Id has the seed value 1 and increment 1 for each inserted record.

Function *Get_ BD_ tree*(R)

1. $R\_tree = \emptyset$
2. $V = \{(a_1 = \text{NULL}, \ldots, a_l = \text{NULL}, c = d_b, lvl = 0, Parent = \text{NULL})\}$
3. $k = 0$
4. $R_k^j = R$
5. **While** $k \geq 0$
6. **If** $\sigma_{lvl=k}(V) \neq \emptyset$
7. $id_j = \min\{\pi_{Id}(\sigma_{lvl=k}(V))\}$
8. $R\_tree = R\_tree \cup \sigma_{Id=id_j}(V)$
9. $(v_{j1}, \ldots, v_{jl}) = \pi_{a_1,\ldots,a_l}(\sigma_{Id=id_j}(V))$

10. $V = V \backslash \sigma_{Id=id_j}(V)$
11. $R^j_{k+1} = R^j_k \backslash (R^j_k \triangleright \triangleleft \pi_b(\sigma_{a_1=v_{j1}\ and\ \ldots\ and\ a_l=v_{jl}}(R^j_k)))$
12. **If** $R^j_{k+1} \neq \emptyset$
    13. $c^j_{k+1} = \max\{|\delta(\pi_b(\sigma_{a_1=v_{i1}\ and\ \ldots\ and\ a_l=v_{il}}(R^j_{k+1})))|$ for $i = 1, \ldots, d_A\}$
    14. $V = V \cup \{(v_{i1}, \ldots, v_{il}, c^j_{k+1}, k+1, id_j)$, where $(v_{i1}, \ldots, v_{il}) \in$
        $\delta(\pi_{a_1,\ldots,a_l}(R^j_{k+1}))$ and $|\delta(\pi_b(\sigma_{a_1=v_{i1}\ and\ \ldots\ and\ a_l=v_{il}}(R^j_{k+1})))| = c^j_{k+1}\}$
    15. $k = k + 1$
16. **Else** $k = k - 1$
17. **Return** *R_tree*

By means of graphical interface the realization of the algorithm allows to select the attributes in the left-hand side and the attribute in the right-hand side of the dependency liable to analyzing (Fig. 2).

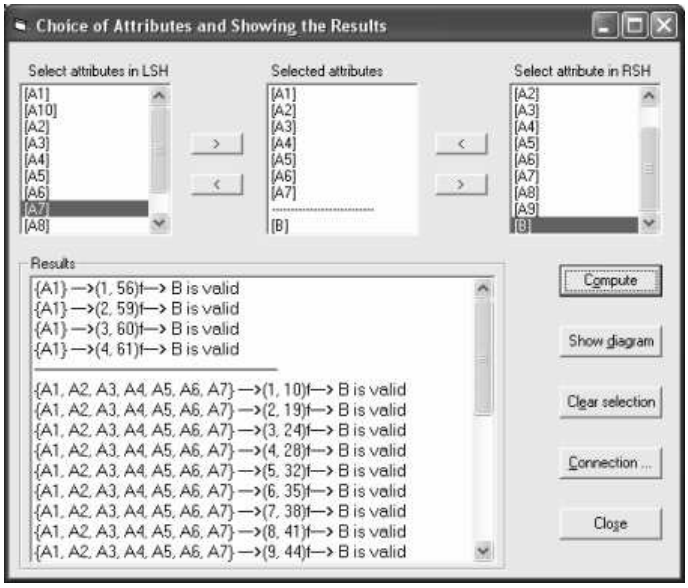

Fig. 2. Results for 1000000 tuples in the relation ($d_{A1} = 8$; $d_X = 404231$, $X = \{A_1, \ldots, A_7\}$)

A database relation with unknown structure is considered. The values of the tuples in the attributes are randomly generated.

Analysis. Since we apply sorting that in the common case has complexity $\Theta(d_A . \log_2 d_A)$, the complexity of the algorithm is $\Theta(p.d_A . \log_2 d_A))$, $1 \leq p \leq d_A$.

The algorithm is realized by using Transact-SQL with the purpose to increase the performance. The executed SQL statements process the datasets to delete the tuples that have to be excluded from the next considerations and computations.

5. Conclusion. Analyzing the dependencies between attributes existing in a given moment allows revealing the valuable knowledge of the structure of the current instance

of a relation. In the present paper the task for discovering all fractional dependencies is considered. An algorithm for finding all fractional dependencies is described.

REFERENCES

[1] J. Demetrovics, G. O. H. Katona, A. Sali. The Characterization of Branching Dependencies. *Discrete Applied Mathematics*, **40** (1992), 139–153.

[2] J. Demetrovics, G. O. H. Katona, A. Sali. Minimal Representations of Branching Dependencies. *Acta Sci. Math. (Szeged)* ,**60** (1995), 213–223.

[3] H. Garcia-Molina, J. D. Ullman, J. Widom. Database Systems: The Complete Book. Williams, 2002.

[4] T. Georgieva. Finding the Branching Dependencies in Random Databases. *Math. and Education in Math.* ,**35** (2006), 258–264.

[5] J. Peneva. Databases – I part. Regalia 6, 2004, (in Bulgarian).

Tsvetanka Lyubomirova Georgieva
"St. St. Cyril and Methodius" University of Veliko Tarnovo
Department of Information Technologies
Veliko Tarnovo, Bulgaria
e-mail: cv.georgieva@uni-vt.bg

НАМИРАНЕ НА ДРОБНИТЕ И ДРОБНИТЕ РАЗКЛОНЕНИ ЗАВИСИМОСТИ В БАЗИ ОТ ДАННИ

Цветанка Л. Георгиева

В настоящата статия е дефинирана дробна зависимост и дробна разклонена зависимост. Разгледани и доказани са някои свойства на тези зависимости. Предложен е алгоритъм за намиране на всички дробни зависимости между дадено множество от атрибути и даден атрибут в релация на база данни.