Tạp chí Tin học và Điều khiển học, T.24, S.1 (2008), 3–11

ON THE DENSE FAMILIES OF DATABASE RELATIONS

VU DUC THI[1], NGUYEN HOANG SON[2]

[1] *Institute of Information Technology, VAST*
[2] *Department of Mathematics, College of Sciences, Hue University*

Abstract. The relational data model was defined by Codd [3] in 1970. Functional dependencies and minimal keys are very important concepts in the relational data model. The dense families of database relations were introduced by Järvinen [5] (2001), who has characterized functional dependencies and minimal keys of relations in terms of dense families. The aim of this paper is to continue investigating some properties of dense families of database relations and their applications. We give necessary and sufficient conditions for an arbitrary family to be $R-$ dense. We prove that for a given relation R the equality set $\mathcal{E}_R$ is an $R-$dense family. We also propose an effective algorithm finding all minimal keys of a given relation R. Finally, some related problems are also studied in this paper.

Tóm tắt. Mô hình dữ liệu quan hệ được đề xuất bởi Codd [3] vào năm 1970. Phụ thuộc hàm và khoá tối tiểu là những khái niệm rất quan trọng trong mô hình dữ liệu quan hệ. Họ trù mật của quan hệ trong cơ sở dữ liệu được giới thiệu bởi Järvinen [5] (2001). Järvinen đặc trưng phụ thuộc hàm và khoá tối tiểu của quan hệ theo quan điểm họ trù mật. Mục tiêu bài báo là tiếp tục nghiên cứu một số tính chất của họ trù mật của quan hệ, và ứng dụng của nó. Chúng tôi đưa ra một số điều kiện cần và đủ để một họ bất kỳ là họ $R-$ trù mật. Chúng tôi chứng minh rằng với một quan hệ cho trước R tập bằng nhau $\mathcal{E}_R$ là một họ $R-$ trù mật. Chúng tôi cũng đưa ra một thuật toán hiệu quả tìm tất cả các khoá tối tiểu của quan hệ cho trước R. Cuối cùng một số vấn đề liên quan cũng được nghiên cứu trong bài báo này.

1. INTRODUCTION

In this section, we begin recalling some main concepts of the theory of relational databases, which can be found in [1, 6, 7].

Let U be a nonempty finite set of *attributes* (e.g. name, age etc). The elements of U will be denoted by $a, b, c, \ldots, x, y, z$, if an ordering on U is needed, by $a_1, \ldots, a_n$. A map *dom* associates with its *domain* $dom(a)$ each $a \in U$. A *relation* R on U is a subset of Cartesian product $\prod_{a\in U} dom(a)$.

We can think of a relation R on U as being a set of tuples: $R = \{h_1, \ldots, h_m\}$,

$$h_i : U \longrightarrow \bigcup_{a\in U} dom(a), h_i(a) \in dom(a), i = 1, 2, \ldots, m.$$

The concept of functional dependency between sets of attributes was introduced by Armstrong [1]. A *functional dependency* (FD for short) is a statement of form $X \rightarrow Y$, where $X, Y \subseteq U$. The FD $X \rightarrow Y$ *holds* in a relation $R = \{h_1, \ldots, h_m\}$ on U if

$$(\forall h_i, h_j \in R)((\forall a \in X)(h_i(a) = h_j(a)) \Rightarrow (\forall b \in Y)(h_i(b) = h_j(b))).$$

We also say that R *satisfies* the FD $X \rightarrow Y$.

This means that the values of the X component of tuples uniquely determine the values of the Y component.

Set F_R be a family of all FDs that holds in R.

Let R be a relation on U and $K \subseteq U$. Then K is called a *minimal key* of R, if it satisfies the following two conditions:

(K1) $K \rightarrow U \in F_R$,

(K2) $\nexists K' \subset K$ such that $K' \rightarrow U \in F_R$.

The subset K which satisfies only (K1) is called a *key* of R.

It is clear that each tuple of a relation must be distinct. Hence, the values of a minimal key can be used to identify tuples.

Note that a relation may have several minimal keys. Denote $\mathcal{K}_R$ the set of all minimal keys of R.

Example 1.1. ([5]) Let us consider the relation R on $U = \{SSN, LNAME, FNAME, DEPT, AGE\}$ as follows:

SNN	$LNAME$	$FNAME$	$DEPT$	AGE
422-11-2320	Benson	Barbara	Mathematics	Young
533-69-1238	Ashly	Dick	Computer Science	Old
489-22-1100	Benson	Mary	Mathematics	Middle-aged
305-61-2345	Davidson	Dick	Computer Science	Old

Table 1. Relation R

It is clear that $\{SNN\}$ is a minimal key of R since two persons cannot have the same value for SNN. Note that $\{LNAME, FNAME\}$ and $\{LNAME, AGE\}$ are also minimal keys of R. Consequently, we have

$$\mathcal{K}_R = \{\{SNN\}, \{LNAME, FNAME\}, \{LNAME, AGE\}\}.$$

In practice, FDs and minimal keys are viewed as properties of *schemes* (a relation scheme s is a pair (U, F), where U is a nonempty finite set of attributes and F is a set of FDs on U); they should hold in very relation of that schemes. Furthermore, minimal keys and dependencies are usually specified by database designers. But as in the previous example, relations may have additional minimal keys which designers are not necessarily aware of. Järvinen has characterized FDs and minimal keys of relations in terms of dense families, and show that generating all minimal keys of a relation R can be reduced to generating all minimal transversals of a simple hypergraph $\min\{\overline{X} : X \in \mathcal{D}, X \neq U\}$, where $\mathcal{D}$ is any $R-$dense family and $\overline{X} = U \backslash X$. Note that an $R-$dense family is a collection of subsets of U, which by applying certain condition induces the set F_R.

In this paper, we investigate the dense families of relations. Our paper is organized as follows. In Section 2, some basic concepts and results about Armstrong axioms and hypergraph theory are given. In section 3, we study some properties of dense families of database relations and their applications. The final section is conclusion.

2. ARMSTRONG AXIOMS AND HYPERGRAPHS

In this section, we present first some related concepts about Armstrong axioms, which may be found in [1, 5, 6, 7].

It is obvious that $F = F_R$ satisfies

(F1) $X \to X \in F$,

(F2) $(X \to Y \in F, Y \to Z \in F) \Rightarrow (X \to Z \in F)$,

(F3) $(X \to Y \in F, X \subseteq V, W \subseteq Y) \Rightarrow (V \to W \in F)$,

(F4) $(X \to Y \in F, V \to W \in F) \Rightarrow (X \cup V \to Y \cup W \in F)$.

A family of FDs satisfying (F1) - (F4) is called an $f - family$ on U.

F_R clearly is an f-family on U. It is known [1] that if F is an arbitrary$f$-family, then there is a relation R on U such that $F_R = F$.

Given a family F of FDs on U, there exists a unique minimal f-family F^+ that contains F. It can be seen that F^+ contains all FDs, which can be derived from F by the rules (F1) - (F4).

Denote $X_R^+ = \{a \in U : X \to \{a\} \in F_R\}$. X_R^+ is called the *closure* of X on R. It is obvious that $X \to Y \in F_R$ if and only if $Y \subseteq X_R^+$.

Let $\mathcal{D} \subseteq \mathcal{P}(U)$ be a family of subsets of U, where $\mathcal{P}(U)$ denotes the *power set* of U, that is, the family of all subsets of U. We define a set $F_\mathcal{D}$ on $\mathcal{D}$ as follows

$$F_\mathcal{D} = \{X \to Y : (\forall A \in \mathcal{D}) X \subseteq A \Rightarrow Y \subseteq A\}.$$

Proposition 2.1. ([5]) *If $\mathcal{D}$ is a family of subsets of a finite set U, then $F_\mathcal{D}$ is an $f-$ family on U.*

Next, we prove the following result.

Proposition 2.2. *Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be two families of subsets of U. If $\mathcal{D}_1 \subseteq \mathcal{D}_2$ then $F_{\mathcal{D}_2} \subseteq F_{\mathcal{D}_1}$.*

Proof. Suppose that $X \to Y \in F_{\mathcal{D}_2}$. Let $A \in \mathcal{D}_1$ such that $X \subseteq A$. Since $\mathcal{D}_1 \subseteq \mathcal{D}_2$, we have $A \in \mathcal{D}_2$. By the definition of $F_{\mathcal{D}_2}$, that is,

$$F_{\mathcal{D}_2} = \{X \to Y : (\forall A \in \mathcal{D}_2) X \subseteq A \Rightarrow Y \subseteq A\},$$

we have $Y \subseteq A$.

Consequently, $X \to Y \in F_{\mathcal{D}_1}$.

The proposition is proved. ■

Hypergraph theory (see, e.g., [2]) is an important subfield of discrete mathematics with many relevant applications in both theoretical and applied computer science.

Now we introduce some basic concepts about hypergraphs which will be needed in the sequel. The concepts and facts given in this section can be found in [2, 4, 6, 7].

Let U be a nonempty finite set and put $\mathcal{P}(U)$ for the family of all subsets of U. The family $\mathcal{H} = \{E_1, E_2, \ldots, E_m\} \subseteq P(U)$ is called a *hypergraph* on U if $E_i \neq \emptyset$ holds for all i (in [2] it is required that the union of E_is is U, in this paper we do not require this condition).

The elements of U are called *vertices*, and the sets $E_1, \ldots, E_m$ are the *edges* of the hypergraph $\mathcal{H}$.

A hypergraph $\mathcal{H}$ is called *simple* if it satisfies

$$\forall E_i, E_j \in \mathcal{H} : E_i \subseteq E_j \Rightarrow E_i = E_j.$$

It can be seen that $\mathcal{K}_R$ is a simple hypergraph.

Let $\mathcal{H}$ be a hypergraph on U. Then $\min(\mathcal{H})$ denotes the set of minimal edges of $\mathcal{H}$ with respect to set inclusion, i.e.,

$$\min(\mathcal{H}) = \{E_i \in \mathcal{H} : \nexists E_j \in \mathcal{H} : E_j \subset E_i\},$$

and $\max(\mathcal{H})$ denotes the set of maximal edges of $\mathcal{H}$ with respect to set inclusion, i.e.,

$$\max(\mathcal{H}) = \{E_i \in \mathcal{H} : \nexists E_j \in \mathcal{H} : E_j \supset E_i\}.$$

It is clear that, $\min(\mathcal{H})$ and $\max(\mathcal{H})$ are simple hypergraphs. Furthermore, $\min(\mathcal{H})$ and $\max(\mathcal{H})$ are uniquely determined by $\mathcal{H}$.

A set $T \subseteq U$ is called a *transversal* of $\mathcal{H}$ (sometimes it is called *hitting set*) if it meets all edges of $\mathcal{H}$, i.e.,

$$\forall E \in \mathcal{H} : T \cap E \neq \emptyset.$$

A transversal T of $\mathcal{H}$ is called *minimal* if no proper subset T' of T is a transversal.

The family of all minimal transversals of $\mathcal{H}$ is called the *transversal hypergraph* of $\mathcal{H}$, and denoted by $Tr(\mathcal{H})$. Clearly, $Tr(\mathcal{H})$ is a simple hypergraph.

Proposition 2.3. ([2]) *Let $\mathcal{H}$ and $\mathcal{G}$ be two simple hypergraphs on U. Then $\mathcal{H} = Tr(\mathcal{G})$ if and only if $\mathcal{G} = Tr(\mathcal{H})$.*

Proposition 2.4. ([2]) *Let $\mathcal{H}$ be a hypergraph on U. Then*

$$Tr(\mathcal{H}) = Tr(\min(\mathcal{H})).$$

The following algorithm allows to finds the family of all minimal transversals of a given hypergraph (by induction).

Algorithm 2.5 ([4])

Input: let $\mathcal{H} = \{E_1, \ldots, E_m\}$ be a hypergraph on U.

Output: $Tr(\mathcal{H})$.

Method:

Step 0. We set $\mathcal{L}_1 := \{\{a\} : a \in E_1\}$. It is obvious that $\mathcal{L}_1 = Tr(\{E_1\})$.

Step q+1. ($q < m$) Assume that

$$\mathcal{L}_q = \mathcal{S}_q \cup \{B_1, \ldots, B_{t_q}\},$$

where $B_i \cap E_{q+1} = \emptyset, i = 1, \ldots, t_q$ and $\mathcal{S}_q = \{A \in \mathcal{L}_q : A \cap E_{q+1} \neq \emptyset\}$.

For each i $(i = 1, \ldots, t_q)$, construct the set $\{B_i \cup \{b\} : b \in E_{q+1}\}$. Denote them by $A^i_1, \ldots, A^i_{r_i} (i = 1, \ldots, t_q)$. Let

$$\mathcal{L}_{q+1} = \mathcal{S}_q \cup \{A^i_p : A \in \mathcal{S}_q \Rightarrow A \not\subset A^i_p, 1 \leqslant i \leqslant t_q, 1 \leqslant p \leqslant r_i\}.$$

Theorem 2.6.([4]) *For every q $(1 \leqslant q \leqslant m)$, $\mathcal{L}_q = Tr(\{E_1, \ldots, E_q\})$, i.e., $\mathcal{L}_m = Tr(\mathcal{H})$.*

It can be seen that the determination of $Tr(\mathcal{H})$ based on our algorithm does not depend on the order of $E_1, \ldots, E_m$.

Remark 2.7.([4]) Denote $\mathcal{L}_q = \mathcal{S}_q \cup \{B_1, \ldots, B_{t_q}\}$, and let l_q $(1 \leqslant q \leqslant m-1)$ be the number of elements of $\mathcal{L}_q$. It can be verified that the worst-case time complexity of our algorithm is

$$\mathcal{O}(|U|^2 \cdot \sum_{q=0}^{m-1} t_q u_q),$$

where $l_0 = t_0 = 1$ and

$$u_q = \begin{cases} l_q - t_q, & \text{if } l_q > t_q; \\ 1, & \text{if } l_q = t_q. \end{cases}$$

Clearly, in each step of our algorithm $\mathcal{L}_q$ is a simple hypergraph. It is known that the size of arbitrary simple hypergraph on U cannot be greater than $C_n^{[n/2]}$, where $n = |U|$. $C_n^{[n/2]}$ is asymptotically equal to $2^{n+1/2}/(\pi.n)^{1/2}$. From this, the worst-case time complexity of our algorithm cannot be more than exponential in the number of attributes. In the case $l_q \leqslant l_m$ $(q = 1, \ldots, m-1)$, it is easy to see that the time complexity of our algorithm is not greater than $\mathcal{O}(|U|^2 \cdot |\mathcal{H}| \cdot |Tr(\mathcal{H})|^2)$. Thus, in these cases this algorithm finds $Tr(\mathcal{H})$ in polynomial time in $|U|, |\mathcal{H}|$ and $|Tr(\mathcal{H})|$. Obviously, if the number of elements of $\mathcal{H}$ is small, then this algorithm is very effective. It only requires polynomial time in $|U|$.

The following proposition is obvious.

Proposition 2.8. ([4]) *The time complexity of finding $Tr(\mathcal{H})$ of a given hypergraph $\mathcal{H}$ is (in general) exponential in the number of elements of U.*

Proposition 2.8 is still true for a simple hypergraph.

3. DENSE FAMILIES

In this section, we first investigate some new properties of dense families of database relations. Note that the notion of dense family of a database relation is defined in [5], as follows:

Let R be a relation on U. We say that a family $\mathcal{D} \subseteq \mathcal{P}(U)$ of attribute sets is $R-$ *dense* (or *dense in* R) if $F_R = F_{\mathcal{D}}$.

The problem is how to find dense families. Järvinen [5] guarantees the existence of at least one dense family. In the sequel, we denote $\mathcal{L}_{F_R}$ simply by $\mathcal{L}_R$, i.e.,

$$\mathcal{L}_R = \{X^+_R : X \subseteq U\}.$$

Proposition 3.1. ([5]) *The family $\mathcal{L}_R$ is $R-$ dense.*

Järvinen presents some fundamental properties of dense families in next proposition.

Proposition 3.2. ([5]) *If $\mathcal{D}$ is $R-$ dense, then the following conditions hold for all $X, Y \subseteq U$:*

(1) $\mathcal{D} \subseteq \mathcal{L}_R$.

(2) $X_R^+ = \bigcap\{A \in \mathcal{D} : X \subseteq A\}$.

(3) $X \to Y \in F_R$ *if and only if* $(\forall A \in \mathcal{D}) X \subseteq A \Rightarrow Y \subseteq A$.

Note that by Proposition 3.1 and Proposition 3.2 (1), $\mathcal{L}_R$ is the greatest $R-$ dense family. Furthermore, Proposition 3.2 (3) implies that each $R-$dense generates the family $\mathcal{L}_R$.

In [5] Järvinen proved the following important result.

Theorem 3.3. ([5]) *Let R be a relation on U. If $\mathcal{D} \subseteq \mathcal{P}(U)$ is $R-$dense, then the following conditions hold:*

(1) *K is a key of R if and only if it contains an element from each set in $\{\overline{A} : A \in \mathcal{D}, A \neq U\}$.*

(2) *K is a minimal key of R if and only if it minimal with respect to the property of containing an element from each set in $\{\overline{A} : A \in \mathcal{D}, A \neq U\}$.*

Now we present another dense family of database relations.

Let $R = \{h_1, \ldots, h_m\}$ be a relation on U, and $\mathcal{E}_R$ the *equality set* of R, i.e.,

$$\mathcal{E}_R = \{E_{ij} : 1 \leqslant i < j \leqslant m\},$$

where $E_{ij} = \{a \in U : h_i(a) = h_j(a)\}$.

Proposition 3.4. ([6]) *The equality set $\mathcal{E}_R$ is $R-$ dense.*

It is easy to see that the dense family $\mathcal{E}_R$ has at most $\frac{m(m-1)}{2}$ elements.

Let $R = \{h_1, \ldots, h_m\}$ be a relation on U, and $\mathcal{N}_R$ be the *nonequality set* of R, i.e.,

$$\mathcal{N}_R = \{N_{ij} : 1 \leqslant i < j \leqslant m\},$$

where $N_{ij} = \{a \in U : h_i(a) \neq h_j(a)\}$.

Let U be a finite set and $\mathcal{P}(U)$ be its power set. For every family $\mathcal{D} \subseteq \mathcal{P}(U)$, the *complement family* of $\mathcal{D}$ is the family $\overline{\mathcal{D}} = \{\overline{A} : A \in \mathcal{D}\}$ on U.

In [6] we also proved the following important result.

Theorem 3.6. *Let R be a relation on U. Then*

(1) $\mathcal{K}_R = Tr(\min(\overline{\mathcal{E}_R}))$.

(2) $\mathcal{K}_R = Tr(\min(\mathcal{N}_R))$.

Now we study some properties of dense families of database relations.

Theorem 3.7. *If $\mathcal{D}$ is $R-$ dense, then*

$$\min(\overline{\mathcal{D}} \setminus \{\emptyset\}) = \overline{\max(\mathcal{E}_R)}.$$

Proof. According to Theorem 3.6 (1), we have $\mathcal{K}_R = Tr(\min(\overline{\mathcal{E}_R}))$, or

$$\mathcal{K}_R = Tr(\overline{\max(\mathcal{E}_R)}). \tag{1}$$

Because $\mathcal{D}$ is $R-$ dense, and by Theorem 3.3, we have $\mathcal{K}_R = Tr(\overline{\mathcal{D}} \setminus \{\emptyset\})$. Furthermore, we also have

$$Tr(\overline{\mathcal{D}} \setminus \{\emptyset\}) = Tr(\min(\overline{\mathcal{D}} \setminus \{\emptyset\})).$$

Hence

$$\mathcal{K}_R = Tr(\min(\overline{\mathcal{D}} \setminus \{\emptyset\})). \tag{2}$$

Combining (1) and (2) gives

$$Tr(\min(\overline{\mathcal{D}} \setminus \{\emptyset\})) = Tr(\overline{\max(\mathcal{E}_R)}).$$

Since $\min(\overline{\mathcal{D}} \setminus \{\emptyset\})$ and $\overline{\max(\mathcal{E}_R)}$ are simple hypergraphs, and according to Proposition 2.3 we have

$$\min(\overline{\mathcal{D}} \setminus \{\emptyset\}) = \overline{\max(\mathcal{E}_R)}.$$

The theorem is proved. ■

From Theorem 3.7, the following corollary is early derived.

Corollary 3.8. *If $\mathcal{D}$ is $R-$ dense, then*

$$\min(\overline{\mathcal{D}} \setminus \{\emptyset\}) = \min(\mathcal{N}_R).$$

In [7] we presented a necessary and sufficient condition for an arbitrary family $\mathcal{D}$ to be $R-$ dense.

Theorem 3.9. *Let R be a relation, $\mathcal{D} \subseteq \mathcal{P}(U)$ be a family of subsets of a U. Then $\mathcal{D}$ is $R-$ dense if and only if for every $X \subseteq U$:*

$$\mathcal{L}_R(X) = \begin{cases} \bigcap\limits_{X \subseteq A} A & \text{if } \exists A \in \mathcal{D} : X \subseteq A, \\ U & \text{otherwise,} \end{cases}$$

where $\mathcal{L}_R(X) = \{a \in U : X \rightarrow \{a\} \in F_R\}$.

Corollary 3.10. *Let R be a relation on U, $\mathcal{D} \subseteq P(U)$ and $U \notin \mathcal{D}$. Then*

(1) *$\mathcal{D}$ is $R-$ dense if and only if $\mathcal{D} \cup \{U\}$ is $R-$dense.*

(2) *$\mathcal{E}_R \cup \{U\}$ is $R-$dense.*

Proof. (1) It is obvious from Theorem 3.9.

(2) The proof is immediate from Proposition 3.4 and (1).

The corollary is proved. ■

We conclude this section by studying some applications of dense families of database relations. Recall that in [7] we gave an effective application of Theorem 3.9, which is an algorithm for finding a cover of FDs of a given relation. In [6] we presented another useful application of Theorem 3.6 (1), which is an algorithm for finding all minimal keys of a given relation R. In many cases our algorithm is more effective. In the same way as in [6], based on Theorem 3.6 (2) we present the following algorithm for finding all minimal keys of a given relation.

Algorithm 3.11. (AM-KEYS)

Input: a relation $R = \{h_1, \ldots, h_m\}$ on U.

Output: $\mathcal{K}_R$.

Method:

Step 1. Construct the nonequality set

$$\mathcal{N}_R = \{N_{ij} : 1 \leqslant i < j \leqslant m\}$$

where $N_{ij} = \{a \in U : h_i(a) \neq h_j(a)\}$.

Step 2. From $\mathcal{N}_R$ compute the family $\min(\mathcal{N}_R) = \{N_{ij} \in \mathcal{N}_R : \nexists N_{pq} \in \mathcal{N}_R : N_{ij} \subset N_{pq}\}$.

Step 3. By Algorithm 2.5 we construct the set $Tr(\min(N_R))$.

By Proposition 2.4, Algorithm 2.5 and Theorem 3.6 (2), we have $\mathcal{K}_R = Tr(\min(\mathcal{N}_R))$. It can be seen that the time complexity of our algorithm is the same time complexity of Algorithm 2.5. In many cases our algorithm is more effective (see Remark 2.7).

It can be seen that, if the number of elements of the nonequality set $\mathcal{N}_R$ is constant, i.e. $|\mathcal{N}_R| \leqslant k$ for some constant k, then the time complexity of finding $\mathcal{K}_R$ of a given relation R is polynomial time [6].

Example 3.12. Let us consider the relation R on U presented in Table 1. Then we have
$N_{12} = U, N_{13} = \{SNN, FNAME, AGE\}, N_{14} = U,$
$N_{23} = U, N_{24} = \{SNN, LNAME\},$
$N_{34} = U,$
$\mathcal{N}_R = \{U, \{SNN, FNAME, AGE\}, \{SNN, LNAME\}\}$, and
$\min(\mathcal{N}_R) = \{\{SNN, FNAME, AGE\}, \{SNN, LNAME\}\}$.
Consequently, $Tr(\min(\mathcal{N}_R)) = \{\{SNN\}, \{LNAME, FNAME\}, \{LNAME, AGE\}\}$.
It is obvious that $\mathcal{K}_R = \{\{SNN\}, \{LNAME, FNAME\}, \{LNAME, AGE\}\}$.

4. CONCLUSION

In this paper, we have studied some new properties of dense families of database relations and their applications.

Let R be a relation on U and $\mathcal{D} \subseteq \mathcal{P}(U)$. We firstly studied some properties of dense families of relation R. That is, we proved that if family $\mathcal{D}$ is $R-$dense, then $\min(\overline{\mathcal{D}} \setminus \{\emptyset\}) = \overline{\max(\mathcal{E}_R)}$ and $\min(\overline{\mathcal{D}} \setminus \{\emptyset\}) = \min(\mathcal{N}_R)$.

We presented some necessary and sufficient conditions for family $\mathcal{D}$ to be $R-$ dense.

We also proved that for a given relation R the equality set $\mathcal{E}_R$ is an $R-$dense family, whose size is at most $\dfrac{m(m-1)}{2}$, where m is the number of tuples in R.

Finally, we presented an effective algorithm for finding all minimal keys of a given relation R. Some related problems are also studied.

REFERENCES

[1] W. W. Armstrong, Dependency structure of database relationship, *Information Processing 74*, North-Holland Pub. Co. (1974) 580–583.

[2] C. Berge, *Hypergraphs: combinatorics of finite sets*, North - Holland, Amsterdam 1989.

[3] E. F. Codd, A relational model for large shared data banks, *Communications of the ACM* **13** (1970) 377–387.

[4] J. Demetrovics, V. D. Thi, Describing candidate keys by hypergraphs, *Computers and Artificial Intelligence* **18** (2) (1999) 191–207.

[5] J. Järvinen, Dense families and key functions of database relation instances, in: Freivalds R. (ed.), Fundamentals of Computation Theory, *Proceedings of the 13th International Symposium, Lecture Notes in Computer Science* **2138** (Springer-Verlag, Heidelberg, 2001) 184–192.

[6] V. D. Thi, N. H. Son, Describing minimal keys by dense families of database relations, *Vietnam Journal of Computer Science and Cybernetics* **21** (2) (2005) 130–136.

[7] V. D. Thi, N. H. Son, Some results related to dense families of database relations, *Acta Cybernetica* **17** (1) (2005) 173–182.

Received on March 13 - 2006