NOVI SAD J. MATH.
VOL. 26, No. 2, 1996, 49–73

LOSSLESS JOINS OF RELATIONAL DATABASE VIEWS

Ivan Luković, Pavle Mogin
Faculty of Technical Sciences, Institute of Industrial Systems
University of Novi Sad
Trg Dositeja Obradovića 7, 21000 Novi Sad, Yugoslavia

Abstract

We proposed an approach to local lossless join preservation, in contrast to the global lossless join preservation of relational database schema. The database designer, using the so-called form types, creates relational database views. These form types contain all information, necessary to design the set of database constraints. We identify criteria to be satisfied if a subschema with lossless join should be designed. It was also shown that for each form type used in the process of designing the database schema subschema, satisfying the lossless join condition, can be generated.

AMS Mathematics Subject Classification (1991): 68P15.
Key words and phrases: form type, object type, functional dependency, embedded join dependency, nonfunctional dependency, canonical cover, relational database schema, synthesis algorithm, subschema design, lossless join, chase process.

1. Introduction

A relational database (*db*) schema design satisfying the Third Normal Form ($3NF$) condition, may be considered as successful only if the conditions

of lossless join preservation and preservation of initial constraint set are fulfilled.

Usually, the *db* schema design process is based on a synthesis algorithm that produces a set of relation schemes in $3NF$. The synthesis algorithm uses a set of functional dependencies (fds), explicitly or implicitly defined by the *db* designer. The *db* schema design is based on the Universal Relation Scheme Assumption (URSA). The initial constraint set preservation is ensured by the synthesis algorithm [1, 2, 8]. However, lossless join preservation is not ensured by the synthesis algorithm itself, so that a scheme constructed from one of equivalent Universal Relation Scheme (URS) keys, should be added to the set of the synthesized relation schemes.

Let (U, C) be an URS, where U is the universal set of the attributes, and C is the set of constraints over U. Let $\Gamma \subseteq C$ denote the set of fds over U. Let

$$S = \{(R_i, \mathcal{K}_i) \in \{1, ..., m\}\}$$

be the set of *db* relation schemes in $3NF$ (obtained by the synthesis algorithm [1, 2, 8]). $\mathcal{K}_i$ is the key set of relation scheme with the attribute set R_i. The lossless join is provided by adding the relation scheme $(R_{m+1}, \{R_{m+1}\})$ to the set S (unless it has already been there), where the attribute set R_{m+1} is one of the keys of the scheme (U, Γ). If $\bowtie (R_1, ..., R_{m+1})$ denotes the join dependency (jd) over (U, Γ), the implication

$$\Gamma \models \bowtie (R_1, ..., R_{m+1})$$

holds, i.e. if $r(U)$ is a relation over URS (U, Γ), then

$$r(u) = \bowtie_{i=1}^{m+1} (\Pi_{R_i}(r)).$$

This fact can be simply proved by applying chase algorithm [8,10].

Suppose, the relation scheme set $\{R_{l_1}, ..., R_{l_k}\}$ is necessary to generate a view on relational *db* schema. With respect to *db* schema design process described above (synthesis algorithm and URS-key scheme adding), for that view it is possible to determine such an embedded $jd \bowtie (R_{l_1}, ..., R_{l_n})(k \leq n)$, that conditions

$$(\forall l_i \in \{l_1, ..., l_n\})(R_{l_i} \in \{R_1, ..., R_{m+1}\}) \quad \text{and}$$

$$(\exists R_j \in \{R_{l_1}, ..., R_{l_k}\})(\forall l_i \subset \{l_1, ..., l_n\})\Big(\bigcup_{i=1}^{n} \Gamma_{l_i} \models R_j \rightarrow R_{l_i}\Big)$$

will hold, where Γ_{l_i} is the set of *fds* of the relation scheme $(R_{l_i}, \mathcal{K}_{l_i}) \in S$. The $jd \bowtie(R_{l_1}, ..., R_{l_n})$ will be satisfied according to the generated *db* schema, i.e. the implication

$$\bigcup_{i=1}^{n}(\Gamma_{l_i}) \models \bowtie (R_{l_1}, ..., R_{l_n})$$

will hold. Thus, for every view, lossless join can be provided, using relation scheme $(R_{m+1}, \{R_{m+1}\})$, if it is necessary.

However, the approach to global lossless join preservation presented here is of little practical value. Any more complex *db* schema has a great number of relation schemes. The relation scheme $(R_{m+1}, \{R_{m+1}\})$ is often very cumbersome, and contains attributes without any intuitive mutual relationship. Therefore, the necessity to maintain the relation over $(R_{m+1}, \{R_{m+1}\})$ is difficult to defend in practice. Also, it is hard to expect the user view corresponding to the URS instance, to appear in practice. Thus, $jd \bowtie(R_1, ..., R_{m+1})$ preservation is not needful.

In this paper we propose an approach to local lossless join preservation at the level of defined view. The main idea is in the following: the *db* designer creates *db* views, defined by form types. Form types should be created with respect to the end-user requests. They carry information about attributes and real world constraints between them. Using these information, the *db* schema can be synthesized. For each previously created form type, such a subschema can be designed that an appropriate embedded jd (defined by subschema) will hold [9, 7, 3, 4].

In the following text, the form type concept will be briefly described. (The concept has been fully considered in [9, 7].) After that, the approach to lossless join preservation corresponding to the form type concept, will be given.

2. Form type

2.1 . Form Type Definition

Let $W(\mathcal{F})$ be the attribute set of a form type named $\mathcal{F}$. Each concrete form (appearance of the form type $\mathcal{F}$) is a tree-structure over the attribute values from $W(\mathcal{F})$. On the other hand, the form type $\mathcal{F}$ consists of the tree structured object types. Each object type contains one, or more attributes

from $W(\mathcal{F})$. At least one key has to be defined for each object type. The object type key enables a unique object appearance identification on the form. The fact that of the object type O_i is direct ascendant of the object type O_j in the form type tree structure, means that zero or more appearances of the object type O_j are related to one appearance of the object type O_i.

Definition 2.1. *A form type is a four-tuple structure* $\mathcal{F}(\mathcal{O}, \psi, C_{\mathcal{F}}, A_{\mathcal{F}}(x))$, *over the set of attributes* $W(\mathcal{F})$, *where:*

$\mathcal{O} = \{(N_i, \mathcal{K}_o^i) |\ i \in \{1, ..., l\}\}$ *is a set of object types,* $N_i \subseteq W(\mathcal{F})$ *is the attribute set and* $\mathcal{K}_o^i = \{\mathcal{K}_j \subseteq N_i |\ j \in \{1, ..., m\}\}$ *is the set of object type keys.*

$\psi \subseteq \mathcal{O} \times \mathcal{O}$ *is a relation defining the form type tree structure over the set of object types.* $(\mathcal{O}_i, \mathcal{O}_j) \in \psi$ *holds if there is an appearance of the object type* $\mathcal{O}_i$, *to which is related more than one appearance of the object type* $\mathcal{O}_j$. *(By the next definition we formalize the terms "object type appearance" and "form type appearance".)*

$C_{\mathcal{F}}$ *is a set of constraints given by the following conditions:*

(1) $$\bigcup_{i=1}^{l} N_i = W(\mathcal{F})$$

(2) $$(\forall \mathcal{O}_i, \mathcal{O}_j \in \mathcal{O})\ (i \neq j) \Longleftrightarrow N_i \cap N_j = \emptyset)$$

(3) $(\forall \mathcal{O}_i \in \mathcal{O})(\forall A, B \in W(\mathcal{F}))((A \in N_i \wedge B \in N_i) \Longleftrightarrow Ibp(\mathcal{F}, A, B))$,

where the predicate $Ibp(\mathcal{F}, A, B))$ *is defined as*

$$(\forall p_{\mathcal{F}} \in SP_{\mathcal{F}})(|A(p_{\mathcal{F}})| = |B(p_{\mathcal{F}})|).$$

$SP_{\mathcal{F}}$ *denotes the set of the all possible appearances of form type* $\mathcal{F}$, *and* $|A(p_{\mathcal{F}})$ *denotes the number of appearances of the attribute* A *on the appearance* $p_{\mathcal{F}} \in SP_{\mathcal{F}}$.

(4) $(\forall O_i, O_j \in \mathcal{O})(((O_i, O_j) \in \psi \wedge (\forall K \in \mathcal{K}_o^j)) \Rightarrow Joo(O_i, O_j, K))$,

where the predicate $Joo(O_i, O_j, K)$ *means that the key* K *uniquely identifies all appearances of the object type* O_j *subordinated to one appearance of the object type* O_i, *and there is no proper subset of* K *with the same property.*

Let O_k be the root object type of the tree structure. Then, the condition

$$(\forall K \in K_o^k)(Jof(O_k, K)), \tag{5}$$

holds, where the predicate $Jof(O_k, K)$ means that the key K uniquely identifies all appearances of the root object type O_k, i.e. uniquely identifies all appearances of the form type $\mathcal{F}$, and there is no proper subset of K with the same property.

$A_{\mathcal{F}}(x)$ is the predicate by which the usage mode of form type $\mathcal{F}$ is defined, where $x \in \{u, r\}$. If a form type is used both for updates and queries the appropriate interpretation of the predicate is $A_{\mathcal{F}}(u)$, but if it is used only for queries over db, the appropriate interpretation is $A_{\mathcal{F}}(r)$. □

The interpretation of the predicate $A_{\mathcal{F}}(x)$ partitions the set of designed form types into the update class (determined by $A_{\mathcal{F}}(u)$) and the query class (determined by $A_{\mathcal{F}}(r)$). The update class is denoted by $\mathcal{SF}_u$, and the query class is denoted by $\mathcal{SF}_r$.

Definition 2.2. *Let a form type $\mathcal{F}$ be given.*

1^0 *Let $O_k(N_k, K_o^k)$ be the root object type of the form type tree structure. The form type appearance, i.e. the appearance of the root object type O_k is a function $p(O_k) : N_k \rightarrow dom(A_{1k}) \times ... \times dom(A_{nk})$, such that*

$$(\forall A_i \in N_k)(p(O_k)(A_i) \in dom(A_i))$$

holds.

2^0 *Let $O_j(N_j, K_o^j)$ be the object type directly subordinated to the object type $O_i(N_i, K_o^i)$ and let $P(O_i) = \{p_l(O_i) | l = 1, ..., n\}$ denote the set of all appearances of the object type O_i within one occurrence of the form type $\mathcal{F}$. An appearance of the object type O_j corresponding to $p_l(O_i) \in P(O)_i)$ is a function $p(O_j, p_l(O_i)) : N_j \rightarrow dom(A_{1j}) \times ... \times dom(A_{nj})$, such that*

$$(\forall A_i \in N_j)(p(N_j, p_l(N_i))(A_i) \in dom(A_i))$$

holds. □

2.2 . Dependencies Implied by Form Types

Object type keys and partial ordering relation ψ bear the information about the set of functional dependencies $F(\mathcal{F})$, defined by the form type $\mathcal{F}$. First,

we observe the root object type in the tree structure. Respecting the interpretation of predicate Jof, all attributes of this object type functionally depend on its keys. Second, we observe a nonroot object type. According to the interpretation of predicate Joo, all its attributes are uniquely identified by the union of keys belonging to the object types on the path from root to the observed object type. Thus, the set of fds, defined by the form type $\mathcal{F}$ is

$$(6) \qquad F(\mathcal{F}) = \{X \rightarrow A | (\exists O_j \in \mathcal{O})(A \in N_j \wedge X \in \mathcal{X}(O_j))\},$$

where $\mathcal{X}(O_j)$ denotes the set of all possible **One-Key-Unions.** A One-Key-Union of the object type O_j is obtained when for each object type contained on the path from the root to the given object type O_j, we choose one and the only one key, and make a union of the so chosen keys.

Object types at the same level of the form type tree structure are mutually independent. This fact suggests the existence of an embedded jd:

$$\overline{J}(\mathcal{F}) = \bowtie (X_1N_1, ..., X_lN_l)$$

expressed by the form type $\mathcal{F}$, such that:

$$(\forall X_iN_i \in \{X_1N_1, ..., X_lN_l\}) | (\exists\, O_i \in \mathcal{O})(X_i \in \mathcal{X}(O_j)).$$

(Remember N_i is the attribute set of the object type O_i).

A $jd\overline{J}(\mathcal{F})$ holds generally only on the proper projection of an URS instance, whereas the *db* design has been accomplished using several different form types. Because of that, it is considered as **embedded** jd.

Let $T(\mathcal{F})$ be a set of attribute sets, such that there is a one-to-one mapping between $T(\mathcal{F})$ and the set of leaves in the form type tree structure. Each element $X_i \in T(\mathcal{F})$ is One-Key-Union the of appropriate leaf O_i :

$$(7) \quad T(\mathcal{F}) = \{X_i \in \mathcal{X}(O_i) | \neg(\exists O_j \in \mathcal{O})((O_i, O_j) \in \psi) \wedge Jkl(X_i, O_i)\}.$$

The meaning of the predicate $Jkl(X_i, O_i)$ is "only one One-Key-Union for the leaf O_i'' : $(\forall X \in \mathcal{X}(O_i))(X \neq X_i \Rightarrow X \notin T(\mathcal{F}))$. The set $T(\mathcal{F})$ will be named the Set of **Carrying Attribute Groups** of form type $\mathcal{F}$.

Let $J(\mathcal{F}) = \bowtie (X_1, ...X_k)$ be a jd, defined by the Set of Carrying Attribute Groups $T(\mathcal{F}) = \{X_1, ..., X_k\}$. It was proved in [7] that $jds\ \overline{J}(\mathcal{F})$ and $J(\mathcal{F})$ are equivalent with respect to $F(\mathcal{F})$:

$$\{\overline{J}(\mathcal{F})\} \cup F(\mathcal{F}) \models J(\mathcal{F}) \wedge \{J(\mathcal{F})\} \cup F(\mathcal{F}) \models \overline{J}(\mathcal{F}),$$

thus, in further considerations we are going to use $J(\mathcal{F})$ instead of $\overline{J}(\mathcal{F})$.

Tree structure partial ordering of any form type, gives the information about nonfunctional dependencies ($nfds$) between appearances of the related object types. Every path from the root to any leaf with the length one or more, defines an nfd. Hence, the set of $nfds$, defined by the form type $\mathcal{F}$ is

$$NF(\mathcal{F} = \{X_i \rightarrow \emptyset | X_i \in T(\mathcal{F})\}. \tag{8}$$

Example 1. Consider the form type $\mathcal{F}a$ are (PURCHASE-ORDER), shown in Figure 1. Meanings of the attributes are as follows: O - order number, D - order item number, A - article identifier, Q - quantity of ordered article and C - customer number. Object types are of rectangular shape. The keys of the corresponding object types are underlined.

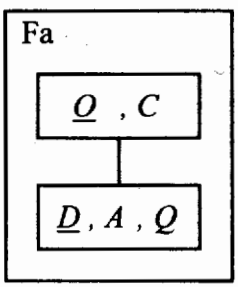

Figure 1.

The set of fds given by $\mathcal{F}_a$ is $F(\mathcal{F}_a) = \{O \rightarrow C, OD \rightarrow A, OD \rightarrow Q\}$, whereas the set of $nfds$ is $NF(\mathcal{F}_a) = \{OD \rightarrow \emptyset\}$. A jd expressed by $\mathcal{F}_a$ is $\overline{J}(\mathcal{F}_a) = \bowtie (OC, ODAQ)$. It is equivalent to trivial jd $J(\mathcal{F}_a) = \bowtie (OD)$. □

Example 2. Consider the form types $\mathcal{F}1$ and $\mathcal{F}2$ shown in Figures 2 and 3. Meanings of the attributes are as follows: A - airplane type identifier, P

- pilot identifier and F - flight identifier.

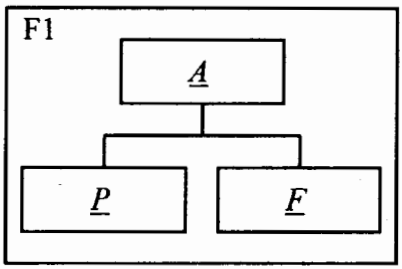

Figure 2.

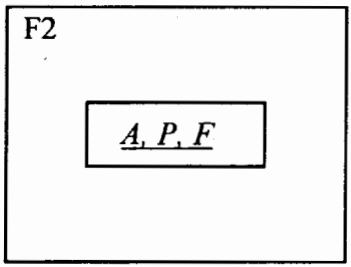

Figure 3.

The form type $\mathcal{F}1$ expresses $nfds$ $AP \rightarrow \emptyset$ (given pilot has got license to fly on the given airplane type), $AF \rightarrow \emptyset$ (given airplane type operates on given flight), and $jd \bowtie (AP, AF)$ (every pilot having license for the given airplane type can fly on all the flights, on which that airplane type operates); the form type $\mathcal{F}2$ expresses an nfd $APF \rightarrow \emptyset$, with the meaning: the given pilot, using the given airplane type, flies on the given flight. □

3. Database schema and closure graph

The proposed *db* design methodology requires that the *db* designer: creates form types, by which the users are going to communicate with IS, and builds a part of the real world constraints into them. After that, the procedure for automatic *db* schema design is applied. Only the update class form types are used by that procedure. Hence, the update class form types should express all relevant real world constraints.

Functional and nonfunctional dependencies used in *db* schema design are derived from the set of update class form types.

A set of fds denoted with Γ, by which the set of *db* relation schemes will be generated, is

$$\Gamma = \bigcup_{\mathcal{F}_i \in \mathcal{SF}_u} (F(\mathcal{F}_i)|_U), \tag{9}$$

where $F(\mathcal{F}_i)|_U$ denotes a projection of the form type fd set onto the *db* schema attribute set U. Namely, the form type $\mathcal{F}$ can have attributes that

do not belong to the *db* schema attribute set U, so it should be noticed that

$$U \subseteq \Gamma = \bigcup_{\mathcal{F}_i \in \mathcal{SF}_u \cup \mathcal{SF}_r} (W(\mathcal{F}_i)).$$

Let us suppose now that there are no form types in $\mathcal{SF}_u$ expressing simultaneously any fd and nfd between the same attributes. If we denote by UNF a set of all $nfds$ expressed by the update-class form types:

$$UNF = \bigcup_{\mathcal{F}_i \in \mathcal{SF}_u} (NF(\mathcal{F}_i)),$$

the following condition will hold:

$$(10) \qquad (\forall (X \to \emptyset,\ V \to Z) \in UNF \times \Gamma^+)(Z \not\subseteq V \Rightarrow VZ \not\subseteq X).$$

In that case, the set of $nfds$, denoted by $N\Gamma$, derived from the update class form types, is

$$(11) \qquad N\Gamma = UNF \setminus lhs(kp(\Gamma)),$$

where $lhs(kp(\Gamma)) = \{X \to \emptyset | (\exists A \in U)(X \to A \in kp(\Gamma))\}$, and $kp(\Gamma)$ is the canonical cover of Γ.

Note that $N\Gamma$ contains only the $nfds$ corresponding to the leaf object types that are not covered by fds from $kp(\Gamma)$.

A *db* relation scheme set satisfying $3NF$, is generated by the synthesis algorithm, using Γ and $N\Gamma$ sets, defined by (9) and (11) [7].

In the sequel, the fact that $(R_k, \mathcal{K}_k) \in S$ is in some formulas shortened as $R_k \in S$.

Definition 3.1. *Relational schema closure graph is the graph* $\mathcal{G} = (S, \rho)$, *where* $S = \{(R_i, \mathcal{K}_i) | i \in \{1, ..., m\}\}$ *is the set of db relation schemes, and the relation* $\rho \subseteq S^2$ *has been given as follows*

$$(12) \qquad \rho = \{(R_i, R_j \in S^2 | R_j \subseteq (R_i)^+_\Gamma \wedge i \neq j \wedge$$

$$\neg(\exists R_k \in S)((k \neq i \wedge k \neq j) \Rightarrow R_j \subseteq (R_k)^+_\Gamma \wedge R_k \subseteq (R_i)^+_\Gamma)\}. \ \square$$

Partial ordering of the *db* relation scheme set S is defined by transitive closure of the relation ρ, denoted by $\tilde{\rho}$. For the sake of simplicity, we are going to use the closure graph $\mathcal{G}$ as a three-tuple structure $\mathcal{G} = (S, \Pi, \mathrm{H})$, where

$\Pi : S \to \mathcal{P}(S)$ is the direct descendant mapping:

$$(\forall R_i \in S)(\Pi(R_i) = \{R_j \in S | (R_i, R_j) \in \rho\}),$$

$H : S \to \mathcal{P}(S)$ is the direct ascendant mapping:

$$(\forall R_i \in S)(\mathrm{H}(R_i) = \{R_j \in S | (R_j, R_i) \in \rho\}).$$

Analogously, the transitive graph $\tilde{\mathcal{G}} + (S, \tilde{\rho})$ can be viewed as a three-tuple $\tilde{\mathcal{G}} = (S, \tilde{\Pi}, \tilde{\mathrm{H}})$.

A closure graph clearly represents relationships between the relation schemes of a normalized db schema. All *fds*, *nfds*, and lossless joins can be easily observed from the closure graph. The set of referential integrity rules

$$SR(\mathcal{G}) = \{R_i[K_j] \subseteq R_j[K_j] | (R_i, R_j) \in \rho\},$$

where $K_j \in \mathcal{K}_j$ is such a key of the relation scheme $(R_j, \mathcal{K}_j)$ that $K_j \subseteq R_i$ holds, is defined by the closure graph.

In the sequel we assume that for each relation scheme from S the whole primary key has been propagated into its direct ascendant nodes as a foreign key, i.e.

$$(\forall (R_i, \mathcal{K}_i) \in S)(\forall R_j \in \mathrm{H}(R_i))(K_i^p \subseteq R_j),$$

where K_i^p is the primary key of the scheme $(R_i, \mathcal{K}_i)$. This assumption simplifies further considerations with no great influence onto generality.

Example 3. Let us consider the form types $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3, \mathcal{F}_4$ and $\mathcal{F}_5$, shown

in Figure 4.

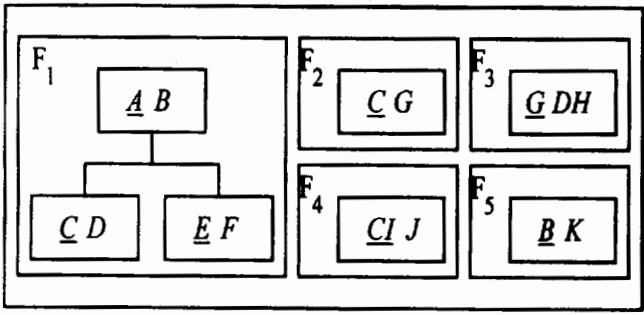

Figure 4.

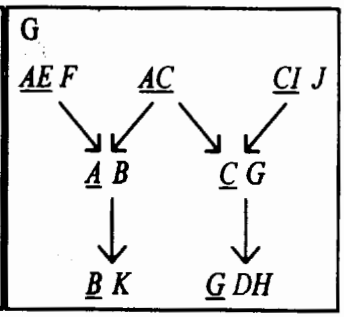

Figure 5.

From the set of fds : $\Gamma = \{A \to B, AC \to D, AE \to F, C \to G, G \to DH, CI \to J, B \to K\}$ we derive the canonical cover $kp(\Gamma) = \{A \to B, AE \to F, C \to G, G \to D, G \to H, CI \to J, B \to K\}$, set of $nfds$ $N\Gamma = \{AC \to \emptyset\}$ and, after the synthesis procedure, we will get the closure graph $\mathcal{G}$, shown in Figure 5. □

4. Subschema design

A subschema of the form type $\mathcal{F}$ is a pair $(P(\mathcal{F}), I)$, where $P(\mathcal{F})$ is a set of relation schemes derived from the set of *db* relation schemes S. I is a set of interrelational integrity rules, derived from the set of *db* schema integrity rules with respect to the set $P(\mathcal{F})$.

In the text to follow, we are going to consider the conditions for generating lossles join subschema with necessary and sufficient relation scheme and attribute sets, regarding the form type $\mathcal{F}$. We are also going to prove the fact that for each update class form type is possible to design a subschema, satisfying these conditions.

In order to identify the conditions subschema design is based on, we will start with a set of relation schemes $P_s(\mathcal{F})$, given by expression

$$P_s(\mathcal{F}) = \{(R_i, K_i) \in S | Min(R_i) \vee Pod(R_i) \vee Ref(R_i)\}, \tag{13}$$

where the predicates $Min(R_i)$ and $Pod(R_i)$ are formalized in the following paragraphs.

The predicate $Min(R_i)$ provides forming of the so-called minimal node set of the form type $\mathcal{F}$. Any relation scheme $(R_i, \mathcal{K}_i) \in S$ (i.e. node of the graph $\mathcal{G}$) satisfies the predicate $Min(R_i)$ iff the following condition is satisfied

$$(\exists X_j \in T(\mathcal{F}))(X_j \in \mathcal{K}_i). \tag{14}$$

Definition 4.1. *A minimal node set of the form type $\mathcal{F}$ is the relation scheme set*

$$P^s_{min}(\mathcal{F}) = \{(R_i, \mathcal{K}_i) \in S | Min(R_i)\}. \quad \square$$

In section 5 we shall prove that for each $X_i \in T(\mathcal{F})$ there is a relation scheme from S whose kay is X_i.

Before we give a formal definition of the predicate $Pod(R_i)$, it is necessary to define the relevant node set of the form type $\mathcal{F}$.

Definition 4.2. *Let the transitive closure graph $\tilde{\mathcal{G}} = (S, \tilde{\Pi}, \tilde{\mathrm{H}})$ and minimal node set $P^s_{min}(\mathcal{F})$ be given. A relevant node mapping of the form type $\mathcal{F}$ $\Omega : P^s_{min}(\mathcal{F}) \to \mathcal{P}(S)$ is defined by the expression*

$$(\forall R_i \in P^s_{min}(\mathcal{F}))(\Omega(R_i) = \{R_j \in \tilde{\Pi}(R_i) \setminus R_i | (R_j \cap W(\mathcal{F})) \setminus K^p_j \neq \emptyset \vee \tag{15}$$

$$(\exists (X_k, K_j) \in \overline{T}(\mathcal{F}) \times \mathcal{K}_j)(X_k = K_j)\}),$$

where $K^p_j \in \mathcal{K}_j$ is the primary key of the scheme $(R_j, \mathcal{K}_j)$, and

$$\overline{T}(\mathcal{F} = \{X_i \subseteq W(\mathcal{F}) | (\exists O_j \in \mathcal{O})(X_i \in \mathcal{X}(\mathcal{O}_j))\}. \quad \square$$

Definition 4.3. *A relevant node set of the form type $\mathcal{F}$ is the set*

$$\mathcal{R}_c(\mathcal{F}) = P^s_{min}(\mathcal{F}) \cup P^s_{rlp}(\mathcal{F}), \tag{16}$$

where $P^s_{rlp}(\mathcal{F})$ is the set of relevant descendant nodes, i.e.

$$P^s_{rlp}(\mathcal{F}) = \{(R_j, \mathcal{K}_j) \in S | ((\exists R_i, \mathcal{K}_i) \in P^s_{min}(\mathcal{F}))(R_j \in \Omega(R_i))\}. \quad \square \tag{17}$$

Relevant nodes are those nodes from the graph $\mathcal{G}$ which contain the attributes from $W(\mathcal{F})$, needed to perform updates into *db* by form type $\mathcal{F}$. The role of other (nonrelevant) nodes, that belong to the subschema of a

given form type $\mathcal{F}$, is to provide lossless join (i.e. to reach the appropriate relevant nodes by selecting one of possible paths from $\mathcal{G}$).

Example 4. Let us consider again the form type $\mathcal{F}_1$, from Example 3. Then, $T(\mathcal{F}_1) = \{AE, AC\}$, $P^s_{min}(\mathcal{F}_1) = \{\underline{AE}F, AC\}$ and $\mathcal{R}_c(\mathcal{F}_1) = \{\underline{AE}F, \underline{AC}, \underline{A}B, \underline{G}DH\}$. □

By means of the predicate $Pod(R_j)$ construction of the necessary descendant node set for the form type $\mathcal{F}$ is defined. Let $SP(R_i, R_k)$ be a set of all paths from the node R_i to node R_k in the graph $\mathcal{G}$. Observe an arbitrary minimal node $R_i \in P^s_{min}(\mathcal{F})$, and any of its relevant descendants $R_k \in \Omega(R_i)$. It is necessary and sufficient to select exactly one path $ip(R_i, R_k) \in SP(R_i, R_k)$ to provide lossless join subschema design. The path $ip(R_i, R_k)$ is selected by applying the following two criteria: **maximal participation of nodes on path** and the shortest path (precisely described in [7]) in the cited order. Thus, in the text below the following assumption holds: for each minimal node and for each its relevant descendant node, exactly one path is selected from the set of all possible paths between that two nodes.

Let SIP be a set of all selected paths from the graph $\mathcal{G}$ obtained using maximal participation and minimal length criteria:

$$SIP = \{ip(R_i, R_k) | R_i \in P^s_{min}(\mathcal{F}) \wedge R_k \in \Omega(R_i) \wedge \tag{18}$$

$$ip(R_i, R_k) \quad selected\text{ - }from \quad SP(R_i, R_k)\}.$$

The relationship *selected-from* means that an $ip(R_i, R_k)$ is exactly one selected path between the nodes R_i and R_k. Let SIC denote a set of all nodes, belonging to a path in SIP :

$$SIC = \{R_j \in S | (\exists ip(R_i, R_k) \in SIP)(R_j \in ip(R_i, R_k))\}, \tag{19}$$

whereas a set of nonrelevant selected nodes is given as: $SIN = SIC \setminus \mathcal{R}_c(\mathcal{F})$.

Arbitrary relation scheme $(R_j, \mathcal{K}_j) \in S$ satisfies the predicate $Pod(R_j)$ iff the following condition is satisfied:

$$(R_j \in P^s_{rlp}(\mathcal{F})) \vee ((R_j \in SIN) \wedge (\exists R_l \in \Pi(R_j) \cap SIC)(K^p_1 \not\subseteq K^p_j)). \tag{20}$$

Definition 4.4. *A necessary descendant node set of the form type $\mathcal{F}$ with respect to the graph $\mathcal{G}$ is the relation scheme set:* $P^s_{pod}(\mathcal{F}) = \{(R_j, \mathcal{K}_j) \in S | Pod(R_j)\}$. □

The necessary descendant node set $P^s_{pod}(\mathcal{F})$ contains relevant descendant nodes and nonrelevant nodes that are contained in selected paths between the starting nodes in $P^s_{min}(\mathcal{F})$ and the ending nodes in $P^s_{rlp}(\mathcal{F})$. It follows from (20) that a nonrelevant node does not belong to $P^s_{pod}(\mathcal{F})$, if for each of its selected direct descendents, the descendent primary key is a subset of the considered nonrelevant node primary key.

Exactly one path in $\mathcal{G}$ between the arbitrary nodes $R_i \in P^s_{min}(\mathcal{F})$ and $R_l \in \Omega(R_i)$, has been selected. However, it follows from (20) and (19) that it is possible to have two or more different paths between certain nodes $R_i \in P^s_{min}(\mathcal{F})$ and $R_k \in \Omega(R_i)$. This situation will happen only if there are two different nodes from the set $(\Omega(R_i) \cap \tilde{\Pi}(R_k)) \setminus R_k$, such that one of two paths is necessary for the first node, while the other is necessary for the second node. Note that R_k belongs to both of the paths.

The predicate $Ref(R_i)$ defines the set of all relation schemes from S, necessary to provide referential integrity checking. These schemes should be added into $Ps(\mathcal{F})$ for the purpose to enable updates by an update class form type $\mathcal{F}$. However, we are not going to consider the predicate $Ref(R_i)$, because it is not important from the point of the lossless join providing. $Ref(R_i)$ has been precisely defined in [7]). Thus, the set $P_s(\mathcal{F})$ is:

$$P_s(\mathcal{F}) = P^s_{min}(\mathcal{F}) \cup P^s_{pod}(\mathcal{F}).$$

Example 5. Reconsider the graph $\mathcal{G}$ from Figure 5 and the form type $\mathcal{F}$ from Figure 4. The necessary descendant node set for $\mathcal{F}$ is $P^s_{pod}(\mathcal{F}) = \{\underline{AB}, \underline{CG}, \underline{GDH}\}$, and

$$P_s(\mathcal{F}) = P^s_{min}(\mathcal{F}) \cup P^s_{pod}(\mathcal{F}) = \{\underline{\underline{AE}F}, \underline{AC}, \underline{AB}, \underline{CG}, \underline{GDH}\}. \ \square$$

In the sequel we shall prove that the set $P_s(\mathcal{F}) = P^s_{min}(\mathcal{F}) \cup P^s_{pod}(\mathcal{F})$ is a necessary and sufficient relation scheme set for lossless join subschema design. The subschema relation scheme set $P(\mathcal{F})$ will be formed by projecting relation schemes from the set $P_s(\mathcal{F})$ onto the set of necessary attributes.

Definition 4.5. *Let $\mathcal{F}$ be a form type with the set of attributes $W(\mathcal{F})$ and $(R_i, \mathcal{K}_i) \in P_s(\mathcal{F})$. The projection of the relation scheme onto the set of relevant attributes of the form type $\mathcal{F}$, denoted as $\Pi_{Y_i}(R_i, \mathcal{K}_i)$, is a relation scheme $(Y_i, \mathcal{K}'_i)$ such that*

$$(21) \qquad Y_i = R_i \cap W(\mathcal{F}) \cup K_p) \wedge \mathcal{K}'_i = \{K^p_i\} \cup \{K_i \in \mathcal{K}_i | K_i \subseteq Y_i\},$$

where $K_p = \bigcup_{R_j \in P_s(\{\})} K_j^p$. □

Note that all attributes of the primary key K_i^p of the relation scheme $(R_i, \mathcal{K}_i) \in P_s(\mathcal{F})$, and all primary key attributes of its direct descendant schemes belong to the set Y_i.

The subschema relation scheme set $P(\mathcal{F})$ of a form type $\mathcal{F} \in \mathcal{SF}_u$ is the set of projections of relation schemes from $P_s(\mathcal{F})$ onto the set of relevant attributes:

$$P(\mathcal{F}) = \{(Y_i, \mathcal{K}'_i) | (\exists (R_i, \mathcal{K}_i) \in P_s(\mathcal{F}))((Y_i, \mathcal{K}'_i) = \Pi_{Y_i}(R_i, \mathcal{K}_i))\}.$$

A set of all **relevant attributes** is the set

$$\Re = \bigcup_{Y_i \in P(\mathcal{F})} Y_i.$$

Other attributes from the set $\left(\bigcup_{R_i \in P_s(\mathcal{F})} R_i\right) \setminus \Re$ are considered as **nonrelevant.**

5. Subschema lossless join

We have presented criteria by which subschema relation scheme set is designed. Thus, we have obtained all necessary elements to formulate the problem of subschema lossless join. The problem of subschema lossless join can be expressed by the following two implicational problems:

$$\{J(\mathcal{F})\} \cup \Gamma \models \bowtie (R_{l_1}, ..., R_{l_n}), \tag{22}$$

where

$$P_s(\mathcal{F}) = \{(R_{l_1}, \mathcal{K}_{l_1}), ..., (R_{l_n}, \mathcal{K}_{l_n})\}$$

is the set of relation schemes forming the subschema, and:

$$\{J(\mathcal{F})\} \cup \Gamma \models \bowtie hd(Y_{l_1}, ..., Y_{l_n}), \tag{23}$$

where $P(\mathcal{F}) = \{(Y_{l_1}, \mathcal{K}'_{l_1}), ..., (Y_{l_n}, \mathcal{K}'_{l_n})\}$ is the relation scheme set of the subschema. The implication $rhd\bowtie (Y_{l_1}, ..., Y_{l_n}) \models \bowtie (R_{l_1}, ..., R_{l_n})$ holds [10], whereas all nonrelevant attributes from $R_{l_i} \setminus Y_{l_i}$ functionally depend on the appropriate primary key $K_{l_i}^p$, which belongs to the set Y_{l_i}. Thus, if condition (23) holds, condition (22) will consequently hold.

Lemma 5.1. *Let Γ and $N\Gamma$ be the sets given by expressions (9) and (11), respectively, $kp(\Gamma)$ be the canonical cover of Γ, and S be the db relation scheme set. Let $\mathcal{F}$ be an $A_{\mathcal{F}}(u)$ form type, having the set $T(\mathcal{F}) = \{X_1, ..., X_k\}$, given by (7), and let $P_s(\mathcal{F})$ be given by (13). Then, the following condition:*

$$(24) \qquad (\forall X_i \in T(\mathcal{F}))(\exists R_j, \mathcal{K}_j) \in P_s(\mathcal{F}))(X_i \in \mathcal{K}_j)$$

holds.

Proof. Observe an arbitrary $X_i \in T(\mathcal{F})$.

a) Suppose, there is a nfd $X_i \to \emptyset \in N\Gamma$. Then, the relation scheme $(X_i, \{X_i\})$ belongs to S. It belongs to the set $P_s(\mathcal{F})$ as well (according to (14)), thus condition (24) holds.

b) If there is no nfd $X_i \to \emptyset \in N\Gamma$, then a nontrivial dependency $X_i \to A$ must exist in the set of fds $F(\mathcal{F})|_U$, and consequently it belongs to Γ (by (9)). Then, one of the following cases holds.

1^0 Let $X_i \to A$ be a left reduced, nonredundant dependency, with respect to Γ. Then, $X_i \to A$ is in $kp(\Gamma)$. Consequently, there exists a relation scheme $(R_j, \mathcal{K}_j) \in S$ such that $A \in R_j$ and $X_i \in \mathcal{K}_j$. According to (14), $(R_j, \mathcal{K}_j)$ is in $P_s(\mathcal{F})$.

2^0 Let $X_i \to A$ be a left reduced, redundant dependency. Thus, it will be omitted from $kp(\Gamma)$. On the other hand, we must have $X_i \to \emptyset \in NF(\mathcal{F})$ because $X_i \in T(\mathcal{F})$ holds, so it will be $X_i \to \emptyset \in N\Gamma$, according to (11). However, it contradicts the b) - assumption $\neg(\exists X_i \to \emptyset \in N\Gamma)$. Therefore, this case is the same as a).

3^0 Let $X_i \to A$ be a partial dependency. According to (10), there is no nontrivial fd between attributes of X_i. It was proved in [7] that $X_i \to A$ is, at the same time, redundant dependency, and thus this case is the same as the previous one. □

Note that previous Lemma has been proved thanks to the set of $nfds$ $N\Gamma$. The existence of $N\Gamma$ is important in resolving the implicational problem (23).

Definition 5.1. *Let an arbitrary $X_i \in T(\mathcal{F})$ and subschema relation scheme set $P(\mathcal{F})$ be given. Let $\mathcal{Y}_i = \mathcal{Y}(X_i)$ be given as follows:*

$$\mathcal{Y}_i = \{Y_j \in P(\mathcal{F}) | Y_j \subseteq (X_i)^+_\Gamma\}.$$

Relevant attribute closure group of given X_i is the set of attributes $G_i = G(X_i)$ defined as:

$$G_i = \bigcup_{Y_j \in \mathcal{Y}_i} Y_j. \ \square$$

Since for each $X_i \in T(\mathcal{F})$ there is in the minimal node set $P^s_{min}(\mathcal{F})$ a scheme $(R_i, \mathcal{K}_i)$ such that $X_i \in \mathcal{K}_i$ holds, then the set $P(\mathcal{F})$ contains an appropriate scheme $(Y_i, \mathcal{K}'_i)$ such that $X_i \subseteq (Y_i)^+_\Gamma$ holds. Whereas X_i is the key of $(R_i, \mathcal{K}_i)$, the condition $Y_i \subseteq (X_i)^+_\Gamma$ holds as well, and consequently $Y_i \subseteq G_i$. The element Y_i for which $(X_i)^+_\Gamma = (Y_i)^+_\Gamma$ holds is named **carrying element of group** $G_i = G(X_i)$. It is the element the procedure of forming the group G_i starts with. The carrying element of group G_i is the minimum of the subgraph $\mathcal{G}_i$ derived from $\tilde{\mathcal{G}} = (S, \tilde{\rho})$ as follows:

$$\mathcal{G}_i = (\mathcal{Y}_i, \mathcal{Y}_i^2 \cap \rho).$$

The fact that for each $X_i \in T(\mathcal{F})$, an appropriate set G_i can be built, (that is, for each X_i there is an appropriate carrying element Y_i) arises from the way on which the db relation scheme set S has been designed, i.e. it is a consequence of Lemma 5.1.

Bearing in mind the definition of the set $G_i = G(X_i)$, we can also consider a closure group relation scheme $(G_i, \{K^p_i\})$, where $K^p_i \in \mathcal{K}_i$ is the primary key of the scheme $(R_i, \mathcal{K}_i) \in P^s_{min}(\mathcal{F})$, for which $X_i \in \mathcal{K}_i$. We are going to denote the set of closure group relation schemes as $\aleph$:

$$\aleph = \{(G_i, \{K^p_i\}) | i \in \{1, ..., k\}\}.$$

Example 6. Reconsider the form type $\mathcal{F}_1$ and canonical cover $kp(\Gamma)$ from Example 3. The subschema relation scheme set is $P(\mathcal{F}) = \{\underline{AE}F, \underline{AC}, \underline{AB}, \underline{C}G, \underline{GD}\}$. Note that the attribute G in the scheme $\underline{C}G$ is necessary, since G is the key of the descendent scheme $\underline{GD}$. The attribute H belongs neither to $W(\mathcal{F}_1)$, nor to any primary key, thus it is omitted from $P(\mathcal{F})$ as

nonrelevant.

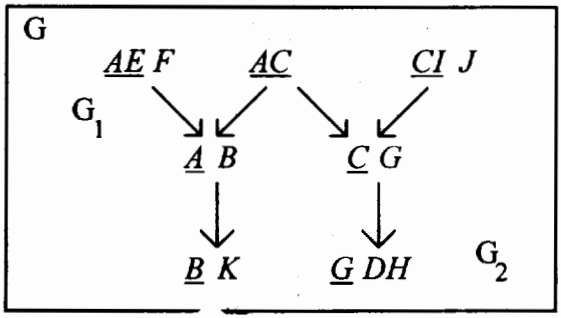

Figure 6.

For the set $T(\mathcal{F}) = \{\underline{AE}, \underline{AC}\}$, the set of closure group relation schemes is $\aleph = \{(ABEF, \{AE\}), (ABCGD, \{AC\})\}$. In Figure 6, the subgraphs $\mathcal{G}_1$ and $\mathcal{G}_2$, corresponding to schemes from $\aleph$, have been bordered by dotted lines. □

The proof of subschema lossless join involves the following global steps. First we should prove for each closure group $G_i \in \aleph$, the lossless join of relation schemes contained into that group (using canonical cover of the set of *fds* $kp(\Gamma)$). Then, the lossless join of all closure groups (i.e. relation schemes from $\aleph$) should be proved. Finally, the implicational problem (23) should be proved on the basis of the *jd* existence inside each closure group, and lossless join of all closure groups.

Lemma 5.2. *Let a db relation scheme set S with the set of fds and subschema relation scheme set $P(\mathcal{F})$ be given. Let $X_i \in T(\mathcal{F})$ and $(G_i, \{K_i^p\}) \in \aleph$ be the closure group relation scheme of X_i with $\mathcal{Y}_i = \{Y_{i_1}, Y_{i_2}, ..., Y_{i_l}\}$. Then the implication*

$$\Gamma \models J_i = \bowtie (Y_{i_1}, Y_{i_2}, ..., Y_{i_l})$$

holds.

Proof. To prove Lemma 5.2, we are going to use the chase algorithm. Thus, we construct the initial tableau $\tau_0(G_i)$ for J_i. τ_0 contains l rows. Each row $l_k \in \tau_0$ ($k \in \{1, ..., l\}$) has distinguished (a) variables for all $A \in Y_k$ of the corresponding $Y_k \in \mathcal{Y}_i$. We denote it as $\tilde{l}_k = Y_k$. Since

$$(\forall Y_k \in \{Y_{i_1}, ..., Y_{i_l}\})(Y_k \subseteq (Y_i)_\Gamma^+)$$

holds, after the chase algorithm terminates, tableau chase $G(\tau_0)$ will have a row containing all distinguished variables, and that row is l_i, i.e. $\tilde{l}_i = G_i$, whereas $G_i = (Y_i)^+_{\Gamma|\mathfrak{R}}$. □

Example 7. Consider the set of *fds* and the set $P(\mathcal{F}_1)$ from Examples 3 and 6, respectively. The set of relevant attributes is $\mathfrak{R} = ABCDEFG$, and projection of Γ onto the attribute set $\mathfrak{R}$ is $\Gamma|_{\mathfrak{R}} = \{A \to B, AE \to F, C \to G, G \to D\}$.

We check the implication: $\Gamma \models \bowtie (AEF, AB)$ by the chase algorithm.

$$A \to B : l_1[A] = l_2[A] \Rightarrow l_1[B] = l_2[B].$$

	t_0	A	B	E	F
l_1	AEF	a	$\underline{a}$	a	a
l_2	AB	a	a		

Now, we check the implication: $\Gamma \models \bowtie (AC, AB, CG, GD)$.

$$C \to G : l_1[C] = l_3[C] \Rightarrow l_1[G] = l_3[G]$$

$$G \to D : l_1[G] = l_4[G] \Rightarrow l_1[D] = l_4[D]$$

$$A \to B : l_1[A] = l_2[A] \Rightarrow l_1[B] = l_2[B].$$

	t_0	A	B	C	D	G
l_1	AC	a	$\underline{a}$	a	$\underline{a}$	$\underline{a}$
l_2	AB	a	a			
l_3	CG			a		a
l_4	GD				a	a

A minimal node of the subgraph $\mathcal{G}_2$, of the group $\mathcal{G}_2$, is $\underline{AC}$, and the row with all distinguished variables is l_1. □

Lemma 5.3. *Consider a form type $\mathcal{F}$, $jd\ J(\mathcal{F}) = \bowtie (X_1, ..., X_k)$, the set of closure groups $\aleph$ and the set of $fds\Gamma$. The implication*

$$\{J(\mathcal{F})\} \cup \Gamma \models J_G = \bowtie (G_1, ..., G_k)$$

holds.

Proof. We are going to use chase algorithm, thus we make the initial tableau $\tau_0(\aleph)$ for $jd\ J_G$. Each row $l_i \in \tau_0$ $(i \in \{1, ..., k\})$ has distinguished variables for all attributes of the corresponding G_i, thus

$$(\forall l_i, l_j \in \tau_0)(l_i[G_i \cap G_j] = l_j[G_i \cap G_j])$$

holds. Since $(\forall X_i \in T(\mathcal{F}))(\exists G_i \in \aleph)(X_i \subseteq G_i)$, the following condition

$$(\forall\ i, j \in \{1, ..., k\})(X_i \cap X_j \subseteq G_i \cap G_j)$$

holds, too. Thus,

$$(\forall\ l_i, l_j \in \tau_0)(l_i[X_i \cap X_j] = l_j[Xi \cap X_j]).$$

According to $jd\ J(\mathcal{F})$, we modify τ_0, by adding a new row l_{k+1}, such that

$$(\forall i \in \{1, ..., k\})(l_{k+1}[X_i] = l_i[X_i]).$$

It follows that the row l_{k+1} has distinguished variables over the set of attributes,

$$\overline{W} = \bigcup_{i=1}^{k} X_i. \quad \text{i.e.} \quad \tilde{l}_{k+1} = \overline{W}.$$

Since $(X_i)^+_\Gamma = (Y_i)^+_\Gamma$, where Y_i is the carrying element of the group G_i, then $(X_i)^+_{\Gamma|\aleph} = G_i$, thus we modify the row l_{k+1} using fds from Γ. Accordingly, the following condition

$$(\forall\ i \in \{1, ..., k\})(l_{k+1}[G_i] = l_i[G_i])$$

holds. Furthermore, it means that , $\tilde{l}_{k+1} = \bigcup_{i=1}^{k} G_i = \Re$, and the row is the one containing all distinguished variables. □

Example 8. Reconsider the form type $\mathcal{F}_1$ from Example 3. The set of closure groups is $\aleph = \{(ABEF, \{AE\}), (ABCGD, \{AC\})\}$. We are going to check the implication $\{\bowtie (AE, AC)\} \cup \Gamma \models \bowtie (ABEF, ABCGD)$.

$$l_1[A] = l_2[A] \Rightarrow l_3[AC] = l_2[AC] \wedge l_3[AE] = l_1[AE].$$

t_0	A	B	C	D	E	F	G
l_1 $ABEF$	a	a			a	a	
l_2 $ABCDG$	a	a	a	a			a
l_3	a	$\underline{a}$	a	$\underline{a}$	a	$\underline{a}$	$\underline{a}$

$$(AE)^+_\Gamma = ABEF \wedge (AC)^+_\Gamma = ABCDG \Rightarrow$$
$$I_3[ABEF] = I_1[ABEF] \wedge I_3[ABCGD] = I_3[ABCGD],$$
$$\text{i.e. } \tilde{I}_3 = ABCDEFG. \square$$

Lemma 5.4. *Let the subschema relation scheme set* $P(\mathcal{F}) = \{(Y_i, \mathcal{K}'_i) \mid i \in \{1, ..., n\}\}$, *the set of closure groups* $\aleph$, *jd* $J_G = \bowtie (G_1, ..., G_k)$, *and the set of*

$$jds\ \mathcal{J} = \{J_i = \bowtie (Y_{i_1}, Y_{i_2}, ..., Y_{i_l}) \mid G_i \in \aleph\}$$

be given. Then, the implication

$$\mathcal{J} \cup \{J_G\} \models \bowtie (Y_1, ..., Y_n)$$

holds.

Proof. We are going to use the chase algorithm, thus we make the initial tableau $\tau_0(\aleph)$ for jd $\bowtie (Y_1, ..., Y_n)$. Let us assume that the rows from $\tau_0(\Re)$ are enumerated in such a way that $(\forall\ l_i \in \tau_0)(\tilde{l}_i = Y_i)$. Therefore, the following condition

$$(\forall\ l_i, l_j \in \tau_0)(l_i[Y_i \cap Y_j] = l_j[Y_i \cap Y_j])$$

holds. Let $J_i = \bowtie (Y_{i_1}, Y_{i_2}, ..., Y_{i_l}) \in \mathcal{J}$ be an arbitrary jd corresponding to the closure group G_i. Since $(\forall Y_k \in \{Y_{i_1}, Y_{i_2}, ..., Y_{i_l}\})(Y_k \in P(\mathcal{F}))$ holds, then τ_0 satisfies the following condition:

$$(\forall\ j, k \in \{1, ..., l\})(l_{i_j}[Y_{i_j} \cap Y_{i_k}] = l_{i_k}[Y_{i_j} \cap Y_{i_k}]).$$

Consequently, regarding $jd\ J_i$, a new row, say l_{n+i}, satisfying the condition:

$$(\forall\, k \in \{1,...,l\})(l_{n+i}[Y_{i_k}] = l_{i_k}[Y_{i_k}])$$

is added into τ_0. Then, since $G_i = \bigcup_{k=1}^{l} Y_{i_k}$, the row l_{n+i} has distinguished variables over G_i, that is $\tilde{l}_{n+1} = G_i$.

During the chase process we expand the initial tableau τ_0 with k new rows, according to all $jds\ J_i \in \mathcal{J}$, since the cardinality of $\mathcal{J}$ is $|\mathcal{J}| = k$. We get $\tau = \tau_0 \cup \{l_{n+1},...,l_{n+k}\}$, such that $(\forall\ i \in \{1,...,k\})(\tilde{l}_{n+1} = G_i)$. Furthermore, the condition:

$$(\forall\, i,j \in \{n+1,...,n+k\})(l_i[G_i \cap G_j] = l_j[G_i \cap G_j])$$

holds. It enables to expand τ with a new row l_{n+k+1}, according to J_G, thus

$$(\forall\, i \in \{n+1,...,n+k\})(l_{n+k+1}[G_i] = l_i[G_i])$$

holds. Therefore , $\tilde{l}_{n+k+1} = \bigcup_{i=1}^{k} G_i$, i.e. $\tilde{l}_{n+k+1} = \Re$, and the final tableau chase $(\tau_0(\Re))$ has a row containing all distinguished variables. □

Example 9. Consider the sets $\aleph$ from Example 8 and $P(\mathcal{F}_1)$ from Example 6, where $\mathcal{G} = \{\underline{AE}BF, \overline{AC}BGD\}$ and $P(\mathcal{F}_1) = \{\underline{AE}F, \underline{AC}, \underline{A}B, \underline{C}G, \underline{G}D\}$. On the basis of $\aleph$ and $P(\mathcal{F}_1)$, the following jds have been given

$$J_G = \bowtie (ABEF, ABCDG) \quad \text{and } \mathcal{J} =$$

$$\{\bowtie (AEF, AB),\ \bowtie (AC, AB, CG, GD)\}.$$

We are going to check the implication

$$\mathcal{J} \cup \{J_G\} \models \bowtie (AEF, AB, AC, CG, GD).$$

by the chase algorithm.

$$\bowtie (AEF, AB):$$

$$l_1[A] = l_2[A] \Rightarrow l_6[AEF] = l_1[AEF] \wedge l_6[AB] = L_2[AB] \Rightarrow \tilde{l}_6 = ABEF$$

$$\bowtie (AC, AB, CG, GD):\ \ (\forall i,j \in \{2,3,4,5\})$$

$$(l_i[X_i \cap X_j] = l_j[X_j \cap X_j]) \Rightarrow l_7[AB] =$$

$$= l_2[AB] \wedge l_7[AC] = l_3[AC] \wedge l_7[CG] =$$

$$= l_4[CG] \wedge l_7[GD] = l_5[GD] \Rightarrow \tilde{l}_7 = ABCDG.$$

$$\triangleright\!\triangleleft (ABEF, ABCDG):$$

$$l_7[AB] = l_6[AB] \Rightarrow l_8[ABEF] = l_6[ABEF] \wedge l_8[ABCDG] =$$

$$= l_7[ABCDG] \Rightarrow \tilde{l}_8 = ABCDEFG. \quad \square$$

t_0		A	B	C	D	E	F	G
l_1	AEF	a				a	a	
l_2	AB	a	a					
l_3	AC	a		a				
l_4	CG			a				a
l_5	GD				a			a
l_6		a	a			a	a	
l_7		a	a	a	a			a
l_8		a	a	a	a	a	a	a

Theorem 5.1. *Let the set of fds* Γ *(9), the form type* $\mathcal{F} \in S\mathcal{F}_u$*, appropriate jd* $J(\mathcal{F})$ *and subschema relation scheme set* $P(\mathcal{F}) = \{(Y_i, K'_i) | i \in \{1, ..., n\}\}$ *be given. Then, the implication*

$$\{J(\mathcal{F})\} \cup \Gamma \models \triangleright\!\triangleleft (Y_1, ..., Y_n)$$

holds.

Proof. Follows from Lemma 5.2, Lemma 5.3, and Lemma 5.4. $\square$

Every designed form type $\mathcal{F}$ expresses some of the real world constraints, represented by the sets $F(\mathcal{F})$, $NF(\mathcal{F})$ and $jd\ J(\mathcal{F})$. The relational *db* schema design is based on update class form types, that is on the constraints, expressed by appropriate sets $F(\mathcal{F})$ and $NF(\mathcal{F})$ and $jd\ J(\mathcal{F})$. According to Theorem 5.1, for each form type used in *db* schema design, a subschema lossless join has been provided. At the same time, the semantics assigned to these form types has been fully preserved.

Every form type $\mathcal{F} \in S\mathcal{F}_u$ can be used for *db* queries. One possible approach to get data form *db* by an $A_{\mathcal{F}}(u)$ form type $\mathcal{F}$ is:

select all relations $r_i(R_i)$ such that $R_i \in P_s(\mathcal{F})$;

make projections of selected relations onto the sets of relevant attributes:

$$(\forall\, R_i \in P_s(\mathcal{F}))(\Pi_{Y_i}(r_i(R_i))),$$

according to relation schemes from $P(\mathcal{F})$;

join $r' = \rhd\lhd_{i=1}^{n} \Pi_{y_i}(r_i)$;

project that join onto the attribute set $W(\mathcal{F})$: $\Pi_W W(r')$.

Theorem 5.1 guarantees that $\bowtie_{i=1}^{n} \Pi_{Y_i}(r_i)$ is lossless join, i.e. r' contains no "false" ("dangling") tuples.

6. Conclusion

In the paper we considered the subschema lossless join problem of an update class form type. The lossless join subschema problem of a query class form type has been fully considered in [7]. On the basis of those considerations, we have developed the algorithms for subschema design and implemented them in IIS*CASE - a CASE tool based on the form type concept, aimed at conceptual data modelling and implementational db schema design. Through the application of IIS*CASE, the presented concepts have been proved in practice.

References

[1] Bernstein, P.A., Synthesizing third normal form relations from functional dependencies, ACM TODS, Vol.1, No.4, Dec. 1976, pp.277-298.

[2] Beeri, C., Bernstein, P.A., Computational problems related the design of normal form relational schemas, ACM TODS, Vol.4, No.1, March, 1979, pp. 30-59.

[3] Choobineh, J., Mannino, V.M., Nunamaker, F.J., Konsynski, R.B., An expert database design system based on analysis of forms, IEEE Transactions on Software Engineering, Vol.14, No.2, Feb. 1988, pp.242-253.

[4] Diet, J., Lochovsky. F., Interactive specification and integration of user views using forms, Proceedings of the Eighth International Conference on Entity-Relationship Approach, Toronto, Canada, 18-20. October, 1989, pp.171-185.

[5] Fagin, R., Mendelzon, O.A., Ullman, D.J., A simplified universal relation assumption and its properties, ACM Transactions on Database Systems, Vol.7, No.3, September 1982. pp.343-360.

[6] Honeyman, P., Sciore, E., A new characterization of independence, ACM-0-89791, 1983.

[7] Luković, I., Automatic relational database subschema design by form types, M.sc Thesis, Faculty of Electrical Science, Belgrade, 18.06.1993.

[8] Maier, D., The Theory of Relational Databases, Computer Science Press, Inc. Rockville, Maryland, 1983.

[9] Mogin. P., Luković, I., Karadžić, Ž., Towards the automatic inference of database constraints, XXXVI Yugoslav conference ETAN, Kopaonik 8-11.06.92, Proceedings, part IX, pp. 19-26.

[10] Paredaens, J., Bra, D.P., Gyssens, M., Gucht, V.D., The Structure of the Relational Database Model, Springer-Verlag, Berlin Heidelberg, 1989.

[11] Ullman, D.J., Principles of Database Systems, Computer Science Press, Inc. Rockville, Maryland, 1982.

Received by the editors November 11, 1994.