An Optimal Algorithm for Computer-Aided Design of Key Type Constraints

Christian Mancaş, Lavinia Crasovschi

Ovidius University, Computer Science Department, 8700 Constanța, Romania
Email: datasis@fx.ro
http://www.datasis.ro/ChristianMancasCV.htm

Abstract. The paper defines entity, relationship, attribute, structural function, and constraint concepts in the framework of the Elementary Mathematical Data Model. A sound and complete algorithm for computer-aided design of key type constraints, which starts with a possibly non-empty set of keys for any object set, and aimed at hopefully completing it with all other existing keys is then introduced and discussed. Finally, this algorithm is proved to be optimal, as it asks db designers the minimum possible number of questions. By simply replacing object sets with relations and mappings with attributes, this proposed algorithm could be used in the Relational Data Model too.

1 Introduction

The *Elementary Mathematical Data Model* (EMDM) is a definite deductive database (db) conceptual modeling tool (see [7] for its formal definition) based on the algebraic theory of sets, relations, and functions, as well as on the first-order predicate calculus with identity. Born from both functional and logical interpretations of the *relational data model* (RDM) [1] and from a rigorous formalization of the *entity-relationship data model* (ERDM) [3], EMDM was continuously refined, mainly by incorporating more and more constraint types (e.g. [4, 5, 8, 9]) and *Datalog¬ programs* [1].

This Section provides a brief introduction into EMDM concepts, emphasizing object sets, mappings, and key type constraints. Section 2 presents and discusses an algorithm for computer-aided design of key type constraints. The algorithm starts with a possibly non-empty set of keys for any object set and smartly generates all other remaining potential keys, which are one by one submitted to the db designer for validation or rejection. The algorithm is then proved to be sound, complete, and optimal (in the sense that it asks db designers the minimum possible number of questions). Finally, Section 3 presents paper conclusions.

1.1 EMDM Schemes

The *scheme* of any db in EMDM is a quadruple ($\mathcal{S}, \mathcal{M}, \mathcal{C}, \mathcal{P}$), where ($\mathcal{S}, \subseteq$) is a nonempty finite poset of *sets*, $\mathcal{M} \subseteq \mathcal{S}^{\mathcal{S}}$ is a nonempty finite set of partially defined *mappings*, $\mathcal{C}$ is a finite set of closed Horn clauses called *constraints*, and $\mathcal{P}$ is a finite set of *Datalog¬ programs*.

In their turn, $\mathcal{S} = \Omega \oplus \mathcal{V} \oplus \varnothing$, where ($\Omega, \subseteq$) is a nonempty finite poset of *object sets* and ($\mathcal{V}, \subseteq$) is a nonempty finite poset of *value sets*; $\mathcal{M} = \mathcal{A} \oplus \mathcal{F}$, where $\mathcal{A} \subseteq \subseteq \mathcal{V}^{\Omega}$ is a nonempty finite set of partial mappings called *attributes* and $\mathcal{F} \subseteq \Omega^{\Omega}$ is a finite set of partial mappings called *structural functions*; for "compact" modeling convenience and historical reasons, both mathematically and db motivated (see mathematical relations and entity-relationship db modeling), a derived concept named *relationship* is provided by EMDM too, thus partioning $\Omega = \mathcal{E} \oplus \mathcal{R}$, where $\mathcal{E}$ is a non-empty *entity sets* and $\mathcal{R}$ a *relationship sets class*; finally, each *Datalog¬* program is a nonempty finite set of open Horn clauses called *inference rules*.

Just like in ERDM, entities model "atomic" objects, which "exists independently of any other object" (i.e. elements of non-Cartesian product sets, e.g. persons, cities, books, bookstores, etc.), while relationships model "compound" ones, "whose existence depends on the existence of other objects" (i.e. elements of Cartesian product sets, e.g. stocks, loans, co-authors, etc.). Note that, unlike ERDM, binary functional relations are modeled in EMDM not as relationships, but as structural functions. For the theory of attributes, structural functions, keys and instances see [4], [9], and [10].

EMDM provides seventeen constraint types: object constraints (i.e. closed Horn clauses over objects); function minimal injectivity, surjectivity, equality, and idempotence; representative systems; binary relations reflexivity, symmetry, antisymmetry, transitivity, and acyclicity; set (strict) inclusion, equality, disjointness, and direct sum, as well as existence and (explicit) domain constraints.

For object and domain constraints, EMDM provides an *object-oriented first-order predicate calculus with identity* [8]. $\forall \omega \in \Omega$, EMDM automatically defines a total one-to-one mapping $\#\omega : \omega \leftrightarrow$ NAT, $\#\omega \in \mathcal{A}$, referred to as the *object identifier* of ω (where NAT denotes the naturals).

Example 1: Let us consider the following EMDM scheme fragment [7]:

Ω = {*SETS, OBJECTS, VALUES, ENTITIES, RELATIONSHIPS, FUNCTIONS, ATTRIBUTES, STRUCT_FUNCTS, CONSTRAINTS, PROGRAMS, INF_RULES, PROG_COMP*}

$\mathcal{V}$ = {NAT, CHAR, BOOL}

$\mathcal{A}$ = {*#S, SetName, SetType, #F, FunctionName, Total, Injective, #C, ConstraintName, ConstraintType, #P, ProgName, #IR, #PC, InfRulePosition*} (all of them totally defined and having obvious domains and codomains)

$\mathcal{F}$ ={*Synonym* : *OBJECTS* $\rightarrow$ *OBJECTS; ObjectId* : *OBJECTS* $\leftrightarrow$ *FUNCTIONS*, *total; Domain* : *FUNCTIONS* $\rightarrow$ *OBJECTS*, *total*; *Codomain* : *FUNCTIONS* $\rightarrow$ *OBJECTS*, *total; Program* : *PROG_COMP* $\rightarrow$ *PROGRAMS*, *total; InferenceRule* : *PROG_COMP* $\rightarrow$ *INF_RULES*, *total*}

$\mathcal{C}$ = {C_1: *SETS* = *OBJECTS* ⊕ *VALUES* ⊕ ∅, C_2: *OBJECTS* = *ENTITIES* ⊕ ⊕*RELATIONSHIPS, C_3*: *FUNCTIONS* = *ATTRIBUTES*⊕*STRUCT_FUNCTS, C_4*: *key FunctionName•Domain*, C_5: *key Program•InferenceRule, C_6*: *key Program•Inf-RulePosition, C_7*:$(\forall x \in FUNCTIONS)(\forall y \in Domain(x))(Total(x) \wedge x(y) \in Codomain(x) \vee \vee \neg Total(x) \wedge x(y) \in Codomain(x) \cup \{\text{NULL}\})$}
(where object constraint C_7 obviously formalizes function definition)

$\mathcal{P}$= {*InclusionTC*={*InclusionTC*(*x*,*y*)←*Inclusion*(*x*,*y*),
InclusionTC(*x*,*y*)← *InclusionTC*(*x*,*z*), *Inclusion*(*z*,*y*)},
EqualityTC={*EqualityTC*(*x*,*y*)←*Equality*(*x*,*y*),
EqualityTC(*x*,*y*)← *EqualityTC*(*x*,*z*), *Equality*(*z*,*y*)}}
(where *Datalog* programs *InclusionTC* and *EqualityTC* are computing transitive closures of inclusions and set equalities respectively). □

[4] and [9] also present and analyze a complete formal description of EMDM (but also of RDM and ERDM) in EMDM, as well as a polynomial algorithm for translating EMDM schemes into RDM *domain-key normal form* (DKNF) ones [1].

1.2 Object Types and Instances

$\mathcal{M}$ is canonically partitioned by the following equivalence relation („*domain equality*"): any two mappings $a,b \in \mathcal{M}$ are equivalent iff *dom*(*a*) = *dom*(*b*). Such a partition is grouping all mappings defined on a same object set; the corresponding quotient set is trivially isomorphic to Ω; $\forall O \in \Omega$, its representative $\hat{o} = \{m \in \mathcal{M} \mid dom(m) = O\} \subseteq \mathcal{M}$ is referred to as the associated (*object*) *type* and is also written as a mapping product:

$$\hat{o} = \prod_{dom(m) = O} m : O \rightarrow \prod codom(m)$$

Example 2: Types corresponding to object sets *SETS* and *FUNCTIONS* from Example 1 are the following: *SETS* = *#S•SetName•SetType*, *FUNCTIONS* = *#F•FunctionName•Total•Injective•Domain•Codomain*. □

Note that any relationship type is implicitly including all of its canonical Cartesian projections (here, obviously, we could have modeled *PROG_COMP* as a relationship having projections *Program* and *InferenceRule*).

The *description* of an object $o \in O \in \Omega$ is $\hat{o}(o)$; the *instance* of object set *O* is the union of the descriptions of all of its objects, i.e. the image of its corresponding type: *Im*($\hat{o}$). The *db instance* is the union of all of the mapping images in $\mathcal{M}$:

$$Im(\mathcal{M}) = \mathcal{M}(\Omega) = \bigcup_{m \in \mathcal{M}} Im(m).$$

1.3 Keys

Any sub-product *K* of a type $\hat{o}$ is said to be a *key* if it is minimally one-to-one (i.e. if it is one-to-one and no proper subset of it is one-to-one); if *K* is one-to-one, then it is a *superkey*. Trivially, any key is a superkey, any one-to-one mapping is a key, any type is a superkey (as instances are sets!), any type has at least one key (in the "worst" case, itself, excluding its object id, is a key!), and may have several keys (at least two, counting its object id too).

Example 3: *SetName, ConstraintName, ProgName*, *FunctionName•Domain*, *Program•InferenceRule*, and *Program•InfRulePosition* from Example 1 are keys. □

Let S be any set and f, g any two mappings totally defined on it; we say that g is *functionally dependent* on f (denoted by $f \rightarrow g$) iff $\mathrm{Ker}f \subseteq \mathrm{Ker}g$ (where *Ker* denotes the *kernel relation*: $\mathrm{Ker}f = \{(x,y) \in S \times S \mid f(x)=f(y)\}$). Equivalently, we say that f is *functionally determining g*. $f \rightarrow g$ is called a *functional dependence* (fd). Obviously, this is a straightforward generalization of the RDM fd definition.

Example 4: Let $S = SETS$, $f = SetName$, and $g = SetType$ from Example 1. Obviously, $SetName \rightarrow SetType$. □

Proposition 1: (*fd characterization*) $f \rightarrow g \Leftrightarrow \exists h: Imf \rightarrow Img$, a unique mapping such that $g = h \circ f$.

Proof: ($\Rightarrow$) Let us define $h : Imf \rightarrow Img$ such that $\forall y \in Imf$, $h(y) = z$, where $y = f(x)$ and $z = g(x)$, and firstly prove that h is well defined:

(h totally defined) According to mapping image definition, $\forall y \in Imf$, $\exists x \in S$, $y=f(x)$; as g is totally defined, $\forall x \in S$, $\exists z \in Img$, $z = g(x) \Rightarrow \forall y \in Imf$, $h(y) = z$.

(h functional) Let $t,u \in Imf$, $t = u$; by h definition, $\exists x,y \in S$ such that $h(t) = v$, where $t=f(x)$, $v=g(x)$, and $h(u)=w$, where $u=f(y)$ and $w=g(y)$; $t=u \Rightarrow f(x)=f(y) \Rightarrow (x,y) \in Kerf$; as $Kerf \subseteq Kerg$, it follows that $(x,y) \in Kerg \Rightarrow g(x) = g(y) \Rightarrow v = w \Rightarrow h(t) = h(u)$.

Let us show that $g = h \circ f$; $\forall x \in S$, denote $z = g(x) \in Img$ and $y = f(x) \in Imf$; according to h definition, $h(y) = z \Rightarrow h(f(x)) = g(x) \Rightarrow h \circ f = g$.

Finally, we prove that h is unique: assume that $\exists h' : Imf \rightarrow Img$ such that $h' \circ f=g$; it follows that, $\forall x \in Imf$, $\exists y \in S$, $x = f(y)$ such that $h'(f(y)) = g(y)$; according to h definition, $h(x) = g(y) \Rightarrow h(x) = h'(x) \Rightarrow h \equiv h'$.

($\Leftarrow$) Assume $f \not\rightarrow g$; $\mathrm{Ker}f \not\subset \mathrm{Ker}g \Rightarrow \exists (x,y) \in \mathrm{Ker}f$ such that $(x,y) \notin \mathrm{Ker}g$, that is $\exists (x,y) \in S^2$, $f(x) = f(y)$ and $g(x) \neq g(y)$; as h functional, $f(x)=f(y) \Rightarrow h(f(x))=h(f(y))$, which, by h definition, implies that $g(x) = g(y)$; this trivially means that the assumption is false and, consequently, $f \rightarrow g$ q.e.d.

Proposition 2: $\rightarrow$ is a pre-order:

(i) $f \rightarrow f$

(ii) $f \rightarrow g \wedge g \rightarrow h \Rightarrow f \rightarrow h$.

Proof:

(i) trivial, as $\mathrm{Ker}f \subseteq \mathrm{Ker}f$, for any mapping f.

(ii) $f \rightarrow g \wedge g \rightarrow h \Rightarrow \mathrm{Ker}f \subseteq \mathrm{Ker}g \wedge \mathrm{Ker}g \subseteq \mathrm{Ker}h \Rightarrow \mathrm{Ker}f \subseteq \mathrm{Ker}h \Rightarrow f \rightarrow h$ q.e.d.

Let $O \in \Omega$ and $f, g \in \Omega^O$ totally defined; if $f \rightarrow g$ for any f, then g is called *minimal*; conversely, if $g \rightarrow f$ for any f, then g is called *maximal*.

Proposition 3: (i) f minimal $\Leftrightarrow |Imf| = 1$

(ii) f maximal $\Leftrightarrow f$ one-to-one.

Proof:

(i)($\Rightarrow$) Assume $|Imf| > 1$; it follows that $\exists x,y \in O$ such that $f(x) \neq f(y)$; let $g \in \Omega^O$ be any mapping such that $g(x) = g(y)$, i.e. $(x,y) \in \mathrm{Ker}g$; as f is minimal, $g \rightarrow f \Rightarrow \mathrm{Ker}g \subseteq \mathrm{Ker}f \Rightarrow (x,y) \in \mathrm{Ker}f \Rightarrow f(x)=f(y)$, which proves that the initial assumption is false.

$(i)(\Leftarrow)$ $|Imf| = 1 \Rightarrow \forall x,y\in O, f(x)=f(y) \Rightarrow \forall x, y\in O, (x,y)\in \mathrm{Ker}f \Rightarrow \mathrm{Ker}f = O^2$; let $g\in \hat{O}$; trivially, $\mathrm{Ker}g \subseteq O^2 = \mathrm{Ker}f \Rightarrow g \rightarrow f \Rightarrow f$ minimal.

$(ii)(\Rightarrow)$ Assume f is not one-to-one; it follows that $\exists x,y\in O, x\neq y$, such that $f(x)=f(y)$, i.e. $(x,y)\in\mathrm{Ker}f$; let $g\in\Omega^O$ such that $g(x)\neq g(y)$; as f is maximal, it follows that $f\rightarrow g \Rightarrow$ $\Rightarrow \mathrm{Ker}f \subseteq \mathrm{Ker}g \Rightarrow (x,y)\in\mathrm{Ker}g \Rightarrow g(x)=g(y)$, which proves that the initial assumption is false.

$(ii)(\Leftarrow)$ f one-to-one $\Rightarrow \mathrm{Ker}f = \{(x,x)|x\in O\}$; it follows that, for any $g\in\Omega^O$, $\mathrm{Ker}f \subseteq$ $\subseteq \mathrm{Ker}g \Rightarrow f\rightarrow g \Rightarrow f$ maximal q.e.d.

Extension of fds from mappings to mapping products is straightforward due to the following Lemma:

Lemma 4: $\mathrm{Ker}f\bullet g = \mathrm{Ker}f \cap \mathrm{Ker}g$.

Proof: $f\bullet g : S \rightarrow Imf \times Img, x \mapsto (f(x),g(x)), \forall x\in S$.

$(\subseteq)$ Let $(x,y)\in\mathrm{Ker}f\bullet g$; $f\bullet g(x) = f\bullet g(y) \Rightarrow (f(x),g(x)) = (f(y),g(y)) \Leftrightarrow f(x) = f(y)$ and $g(x) = g(y) \Rightarrow (x,y)\in\mathrm{Ker}f$ and $(x,y)\in\mathrm{Ker}g \Rightarrow (x,y)\in\mathrm{Ker}f \cap \mathrm{Ker}g$.

$(\supseteq)$ Let $(x,y)\in\mathrm{Ker}f \cap \mathrm{Ker}g \Rightarrow f(x)=f(y) \wedge g(x)=g(y) \Rightarrow (f(x),g(x))=(f(y),g(y)) \Rightarrow$ $(x,y)\in\mathrm{Ker}f\bullet g$ q.e.d.

Proposition 5: (i) $f\bullet g \rightarrow f$; $f\bullet g \rightarrow g$

(ii) $h\rightarrow f \wedge h \rightarrow g \Rightarrow h \rightarrow f\bullet g$.

Proof: (i) By Lemma 4, $\mathrm{Ker}f\bullet g = \mathrm{Ker}f \cap \mathrm{Ker}g$; $\mathrm{Ker}f \cap \mathrm{Ker}g \subseteq \mathrm{Ker}f \Rightarrow f\bullet g \rightarrow f$; similarly, $\mathrm{Ker}f \cap \mathrm{Ker}g \subseteq \mathrm{Ker}g \Rightarrow f\bullet g \rightarrow g$.

(ii) $h\rightarrow f \wedge h \rightarrow g \Rightarrow \mathrm{Ker}h \subseteq \mathrm{Ker}f \wedge \mathrm{Ker}h \subseteq \mathrm{Ker}g \Rightarrow \mathrm{Ker}h \subseteq \mathrm{Ker}f \cap \mathrm{Ker}g = \mathrm{Ker}f\bullet g \Rightarrow$ $h\rightarrow f\bullet g$ q.e.d.

Theorem 6: (i) $f_1\bullet\ldots\bullet f_n \rightarrow f_i, \forall i\in\{1,\ldots,n\}$

(ii) $g\rightarrow f_i, \forall i\in\{1,\ldots,n\} \Rightarrow g \rightarrow f_1\bullet\ldots\bullet f_n$.

Proof: (i) By Lemma 4, $\mathrm{Ker}f_1\bullet\ldots\bullet f_n = \mathrm{Ker}f_1\cap\ldots\cap\mathrm{Ker}f_n$; as $\mathrm{Ker}f_1\cap\ldots\cap\mathrm{Ker}f_n \subseteq$ $\subseteq\mathrm{Ker}f_i\ \forall i\in\{1,\ldots,n\} \Rightarrow f_1\bullet\ldots\bullet f_n \rightarrow f_i, \forall i\in\{1,\ldots,n\}$.

(ii) $g\rightarrow f_i, \forall i\in\{1,\ldots,n\} \Rightarrow \mathrm{Ker}g\subseteq\mathrm{Ker}f_i, \forall i\in\{1,\ldots,n\} \Rightarrow \mathrm{Ker}g \subseteq \mathrm{Ker}f_1\cap\ldots\cap\mathrm{Ker}f_n \Rightarrow$ $\Rightarrow g \rightarrow f_1\bullet\ldots\bullet f_n$ q.e.d.

By notational abuse, if, for example, $X = f_1\bullet f_2\bullet\ldots\bullet f_k$, we also denote by X the set $\{f_1,f_2,\ldots,f_k\}$ and/or the string $f_1f_2\ldots f_k$.

Proposition 7: $\forall\hat{O}, \forall K\subseteq\hat{O}, K$ key $\Rightarrow \forall m\in\hat{O}, K\rightarrow m$.

Proof: By Proposition 3(ii) K is maximal $\Rightarrow \forall m\in\hat{O}, K\rightarrow m$ q.e.d.

Corollary 8: $\forall\hat{O}, \forall K\subseteq\hat{O}, K$ key $\Rightarrow K\rightarrow\hat{O}$.

Proof: By Proposition 7, $\forall m\in\hat{O}, K\rightarrow m$; by Theorem 6$(ii)$, $K\rightarrow\hat{O}$ q.e.d.

Example 5: In Example 1, if O = *FUNCTIONS* and K = *FunctionName•Domain*, *FunctionName•Domain→#F•FunctionName•Total•Injective•Domain•Codomain*.

Corollary 9: If K is a key of $\hat{O}$, then $\forall S\in\hat{O} - K, S\neq\varnothing$, KS cannot be a key of $\hat{O}$.

Proof: K key $\Rightarrow K$ is minimal $\Rightarrow KS$ is not minimal $\Rightarrow KS$ cannot be a key of $\hat{O}$.

Theorem 10: K key $\wedge K'\rightarrow K \Rightarrow K'$ one-to-one.

Proof: Assume K' is not one-to-one; it follows that $\exists x,y\in O, x\neq y$, such that $K'(x)=$ $=K'(y)$, i.e. $(x,y)\in\mathrm{Ker}K'$; $K'\rightarrow K \Rightarrow \mathrm{Ker}K'\subseteq\mathrm{Ker}K \Rightarrow (x,y)\in\mathrm{Ker}K \Rightarrow K(x)=K(y) \Rightarrow K$ is not one-to-one, which proves that the initial assumption is false q.e.d.

Any two mappings f, g are said to be *functionally equivalent* (denoted $f \leftrightarrow g$) iff $f \to g$ and $g \to f$; obviously, $\leftrightarrow$ is an equivalence relation. For example, all minimal mappings are equivalent, as well as all maximal ones. Trivially, $f \leftrightarrow g \Leftrightarrow \mathrm{Ker}f = \mathrm{Ker}g$.

Proposition 11: (*i*) $(f\bullet g)\bullet h \leftrightarrow f\bullet(g\bullet h)$
(*ii*) $f\bullet g \leftrightarrow g\bullet f$
(*iii*) $f\bullet f \leftrightarrow f$.

Proof: (*i*) By Lemma 4, $\mathrm{Ker}(f\bullet g)\bullet h = \mathrm{Ker}f\bullet g \cap \mathrm{Ker}h = \mathrm{Ker}f \cap \mathrm{Ker}g \cap \mathrm{Ker}gh =$ $=\mathrm{Ker}f \cap \mathrm{Ker}g\bullet h = \mathrm{Ker}f\bullet(g\bullet h) \Leftrightarrow (f\bullet g)\bullet h \leftrightarrow f\bullet(g\bullet h)$

(*ii*) $\mathrm{Ker}f\bullet g = \mathrm{Ker}f \cap \mathrm{Ker}g = \mathrm{Ker}g \cap \mathrm{Ker}f = \mathrm{Ker}g\bullet f$

(*iii*) Trivial, as $\mathrm{Ker}f\bullet f = \mathrm{Ker}f \cap \mathrm{Ker}f = \mathrm{Ker}f$ q.e.d.

Proposition 12: $\forall K,K'$ keys, $K \leftrightarrow K'$.

Proof: ($\Rightarrow$) Let $k' \in K'$; by Proposition 3(*ii*), K maximal $\Rightarrow K \to k'$; by Theorem 6(*ii*), it follows that $K \to K'$.

($\Leftarrow$) Let $k \in K$; by Proposition 3(*ii*), K' maximal $\Rightarrow K' \to k$; by Theorem 6(*ii*), it follows that $K' \to K$ q.e.d.

In [9] it is proved that the mappings quotient set with respect to $\leftrightarrow$ is a finite complete lattice, whose first element is the class of minimal mappings, last element is the keys class, and $\sup\{\hat{f}, \hat{g}\} = \widehat{fg}$. Also shown is that any type is in DKNF [6].

Let O be a type and $\mathrm{card}(O\backslash\{\#O\}) = n \geq 0$.

Proposition 13: $\hat{o}$ has $(2^n - 2)$ proper subsets.

Proof: $\hat{o}$ powerset has $C_n^0 + C_n^1 + \ldots + C_n^{n-1} + C_n^n = 2^n$ elements, out of which 2 are not proper: $\varnothing$ and $\hat{o}$ itself q.e.d.

Proposition 14: $\hat{o}$ has at most $C_n^{\left[\frac{n}{2}\right]}$ simultaneous keys.

Proof: (the following proof is inspired by the Sperner lemma proof from [2])

A *chain* of length n made up of subsets of $\hat{o}$ is a sequence $M_1 \subset M_2 \subset \ldots \subset M_{n-1} \subset \hat{o}$ such that $|M_i| = i$, for $i=1,2,\ldots,n-1$. Trivially, the number of chains of length n that can be constructed in $\hat{o}$ is $n!$. Let $X \subset \hat{o}$, $|X| = r$, $r < n$; the number of chains of length n that include X and have form $M_1 \subset \ldots \subset M_{r-1} \subset X \subset M_{r+1} \subset \ldots \subset \hat{o}$ is equal to $r!(n-r)!$.

Let $1 \leq i,j \leq n$, $i \neq j$; by definition, X_i and X_j might simultaneously be keys iff $X_i \not\subset X_j$ and $X_j \not\subset X_i$. Obviously, if X_i and X_j are not $\subseteq$-comparable, then any chain containing X_i is different of any chain containing X_j.

Denoting by p the number of possible simultaneous keys and by $|M_i| = n_i$, it follows that $\sum_{i=1}^{p} n_i!(n-n_i)! \leq n!$. As $\max_{1 \leq n_i \leq n} C_n^{n_i} = C_n^{\left[\frac{n}{2}\right]}$, this implies that:

$$\frac{p}{C_n^{\left[\frac{n}{2}\right]}} \leq \sum_{i=1}^{p} \frac{1}{C_n^{n_i}} \leq 1 \Rightarrow \max p \leq C_n^{\left[\frac{n}{2}\right]}.$$

Conversely, by considering $\hat{o}$ subsets family having $m = \left[\frac{n}{2}\right]$ elements, it can be easily proved that $\max p \geq C_n^{\left[\frac{n}{2}\right]}$. Consequently, it follows that $\max p = C_n^{\left[\frac{n}{2}\right]}$ q.e.d.

1.4 Relationships

$\forall A \in \mathcal{A}$, $A = (S_A, G_A)$, where $S_A = (\mathcal{MS}_A; \mathcal{MK}_A)$ is its *scheme* (with $\mathcal{MS}_A \subseteq \Omega$ being a nonempty set of *sorts*, whereas $\mathcal{MK}_A \subseteq \mathcal{P}(\mathcal{MS}_A)$ a nonempty set of *structural keys*) and the *graph* G_A is a nonempty subset of the Cartesian product of the sets in $\mathcal{MS}_A$ (generally omitted in standard EMDM notation).

Note that in EMDM all structural keys of a relationship are declared together with that relationship, between accolades, following its sort names set [4].

Example 6: *PROG_COMP* from Example 1 could have been modeled as a relationship having the following scheme: *PROG_COMP* = (*PROGRAMS, INF_RULES*; {*PROGRAMS, INF_RULES*}), where:

$\mathcal{MS}_A$ = {*PROGRAMS, INF_RULES* }

$\mathcal{MK}_A$ = {{*PROGRAMS, INF_RULES* }}.

By common notational abuse, any of its graph instances are also denoted by *PROG_COMP*. □

Note that for mathematical relations $\mathcal{MS}_A$ is an ordered list of sets, while in EMDM it is a set of set names; this means that if a same set is required several times as a relationship sort, distinct synonyms should be provided for each of its instances.

Example 7: Let *SET_EQUALITIES* $\subseteq$ *CONSTRAINTS*; if we are modeling it as a relationship, we need at least one synonym for *SETS*: let *EQUAL_SETS* $\in$ *SETS, Synonym*(*EQUAL_SETS*) = *SETS*; then *SET_EQUALITIES* = (*SETS, EQUAL_SETS*; {*SETS, EQUAL_SETS*}) . □

Mathematically speaking, relationships are elements of the Cartesian products quotient set with respect to their base sets permutation equivalence relation (see [4]).

Let $A \in \mathcal{A}$ be any relationship set and n its *arity* (i.e. the cardinality of $\mathcal{MS}_A$); let $S_1, S_2, ..., S_n$ be the sorts of A, and $\#S_1, \#S_2, ..., \#S_n$ their corresponding object identifiers. Let us canonically extend $\#S_i$, $1 \leq i \leq n$, from their sorts to the relationship graph G_A (thus obviously obtaining the canonical projections of the corresponding Cartesian product): $A_i : G_A \rightarrow \text{NAT}$, $A_i(o_1, ..., o_i, ..., o_n) = \#S_i(o_i)$, $\forall o_k \in S_k$, $1 \leq i, k \leq n$. Note that, generally, they are not one-to-one anymore.

However, their product is always one-to-one, $\forall A \in \mathcal{A}$, but generally not a *key* (i.e. not minimally one-to-one). When, however, also minimal, it constitutes the only *structural key* of the corresponding relationship type object set.

Syntactically, in such simple cases where the whole projections product is the structural key, EMDM provides an abbreviation, by implicitly assuming it and letting db designers omit $\mathcal{MK}_A$ (e.g. *PROG_COMP* = (*PROGRAMS*, *INF_RULES*)).

[10] proves that relationships are not fundamental, but merely derived concepts and that, from the data modeling point of view, the only "interesting" relations are the binary functional ones.

2 An Algorithm for Computer Aided Design of Key Type Constraints

Based on results presented in Section 1, we devised the following algorithm for assisting db designers in specifying all keys of an object set *O*:

Algorithm 15:

In: $K \setminus \{\#O\}$, the set of "semantic" keys of object set *O* (i.e. not including its object id), having $|K|=k \geq 0$.

Out: $K' \supseteq K$, the set of all "semantic" keys of O, $|K'|=k'$

Algorithm:

```
$K' = K$; $k' = k$;
if $\hat{O} \notin K'$
 repeat until all $C_n^{\left\lceil \frac{n}{2} \right\rceil}$ keys have been asserted
   or all $2^n$-2 subsets have been analyzed
    $i$=1
    repeat until i=n
      repeat for all p products containing i mappings
        if $p \notin K'$ and p does not include any key
         then ask db designer whether p is a key
          if answer is yes
            then $K' = K' \cup \{p\}$;
              $k'=k'+1$;
          endif;
        endif;
      endrepeat;
      $i=i+1$;
    endrepeat;
 endrepeat;
endif;
if $K'==\emptyset$          // no key was asserted
 then $k'$=1;
      $K' = \prod_{dom(a)=O} a: O \setminus \{\#O\} \rightarrow codomain(a)$
endif;
```

Abstract program:

$K' = K$; $k' = k$; $kmax = C_n^{\left\lceil \frac{n}{2} \right\rceil}$; $i = 1$;

```
if $\hat{O} \notin K'$
 then
  dowhile $i \leq n - 1$ and $k' < kmax$
   $j = 1; jmax = C_n^i$;
   dowhile $j \leq jmax$
    generate $T_{i,j}$, the $j$-th subset with $i$ elements of $\hat{O}$;
    if $T_{i,j} \notin K'$
     then $l = 1$; $superkey = false$;
      dowhile $l \leq k'$ and not superkey
       if $K'(l) \subseteq T_{i,j}$
         then $superkey = true$;
         else $l = l + 1$;
       endif;
      enddowhile;
      if not superkey
       then ask db designer if $T_{i,j}$ is a key;
           if answer is yes
            then $k' = k' + 1$;
                 $K'(k') = T_{i,j}$;
           endif;
      endif;
     endif;
    $j = j + 1$;
   enddowhile;
   $i = i + 1$;
  enddowhile;
end if;
if $k'==0$ // there is no asserted key other than object id #O
     then $k' = 1$;
     $K' = \prod_{dom(a)=O} a:O\setminus\{\#O\} \rightarrow codomain(a)$
endif;
endAlgorithm 15; □
```

Example 8: If Algorithm 15 were to be applied to an object set $\hat{O} = A \bullet B \bullet C$, assuming that it starts with an empty key set and db designer answers are all *no*, it would obviously generate the following sequence: *A*, *B*, *C*, *AB*, *AC*, *BC*; finally, it would then automatically add the default key *ABC*.

If, for example, answer were *yes* for *A*, then db designer would not be asked anymore whether *AB* and *AC* are keys, and so on. □

Theorem 17 (*Algorithm* 15 *characterization*)

(*i*) Algorithm 15 complexity is $O(2^n C_n^{\left[\frac{n}{2}\right]})$

(*ii*) Algorithm 15 is sound and complete.

(*iii*) Algorithm 15 is optimal, i.e. it asks db designers the minimum possible number of questions.

Proof:

(*i*) Trivial, as it has three embedded loops, all of them finite (hence it never loops infinitely) and the first one depending on *n*, but together with the second taking at most $C_n^1 + ... + C_n^{n-1} = 2^n$-2 steps, while the third one is taking at most $C_n^{\left\lceil \frac{n}{2} \right\rceil}$ steps.

(*ii*)(*Completeness*) All possible keys are considered, as the algorithm is generating all 2^n-2 subsets of any type having cardinal *n*, or all of the maximum $C_n^{\left\lceil \frac{n}{2} \right\rceil}$ number of possible simultaneous keys.

(*Soundness*) Moreover, as each one of them is first checked to ensure that it is not a superkey and only then db designer is asked whether it is a key, it follows that the algorithm is computing only possible remaining keys.

(*iii*) In the worst case (that is if *K* is initially void and db designer answers are all *no*) there should obviously be 2^n-2 questions asked (both the empty set and $\hat{O}$ are excluded), which is exactly the number of questions asked by Algorithm 15, as no possible key is processed more than once. In all other cases, as Algorithm 15 starts with atomic keys and is adding in each outmost loop step one more mapping (up to at most *n*-1), it follows that each key is eliminating the maximum possible number of further questions q.e.d.

3 Conclusions

The paper presents the EMDM notions of objects, attributes, structural functions, and keys. An algorithm for assisting db designers in specifying all of and only the possible keys of any object type is then introduced. Moreover, it is proved that this algorithm is optimal, in that it asks the minimum number of questions possible.

Obviously, this algorithm can guarantee only syntactic correctness of keys; the semantic one is the responsibility of db designers.

We believe that Algorithm 15 is worthwhile considering by any db designer, even if a non-empty set of semantic keys has already been asserted for any object set: running it ensures rechecking, in order not to forget any of them.

Note that, even if it was introduced in the EMDM framework, this algorithm should be embedded in any other data model, as keys are probably the most important constraints. For example, in order to use it in RDM, one should simply replace object sets with relations and mappings with attributes.

Algorithm 15 is implemented in *MatBase* [4, 8, 9], a prototype knowledge-base management system based on EMDM. Moreover, *MatBase* is also incorporating another similar algorithm for helping db designers in asserting relationships structural keys [11].

Trivially, this algorithm is not optimal when considering the number of steps (on the contrary, it is rather a "brute force" type one!). Further research will be done in order to optimize it in this respect too, by not generating subsets that include existing keys. Anyhow, even this exponential one runs fast enough in *MatBase* as type cardinalities are generally under 10 and in worst cases under 100.

Acknowledgements

This research was jointly sponsored by BluePhoenix Solutions, Cary, NC (www.bluephoenixsolutions.com) and Datasis ProSoft srl, Bucharest, Romania (www.datasis.ro).

References

1. Abiteboul S., Hull R., Vianu V.: *Foundations of Databases*. Addison-Wesley (1995)
2. Lubell D.: A Short Proof of Sperner's Lemma. *J. Combinatorial Theory*, 1 (1966) 299
3. Chen P.P.: The Entity-Relationship Model: Toward a Unified View of Data. ACM Trans. on Database Syst. 1(1) (1976) 9-36
4. Mancas C.: Conceptual Data Modeling. Ph.D. Thesis, Bucharest "Polytechnic" University (1997)
5. Mancas C.: On Modeling Closed Entity-Relationship Diagrams in an Elementary Mathematical Data Model., *Proceedings 6th East-European Conference on Advances in Databases and Information Systems (ADBIS),* Bratislava, Slovakia, (2002) 165-174.
6. Mancas C.: On Modeling the Relational Domain-Key Normal Form Using an Elementary Mathematical Data Model. *Proceedings IASTED Int. Conf. on Software Engineering and Applications (SEA),* Cambridge, MA (2002) 767-772
7. Mancas C: On Knowledge Representation Using an Elementary Mathematical Data Model. *Proceedings IASTED Int. Conf. on Information and Knowledge Sharing (IKS),* St. Thomas, V.I. (2002) 206-211
8. Mancas C., Dragomir S., Crasovschi L.: On Modeling First Order Predicate Calculus Using the Elementary Mathematical Data Model in *MatBase* DBMS. *Proceedings 21st IASTED Int. Conf. on Applied Informatics (AI),* Innsbruck, Austria (2003) 1197-1202
9. Mancas C.: Databases and Data Structures. Volume I: Conceptual Data Modeling and Querying. To be published by Ovidius University Press (2003)
10. Mancas C., Dragomir S.: On the Equivalence of Functional and Entity-Relationship Data Modeling. Paper submitted at IASTED SEA2003 International Conference, to be held November 2003 at Marina del Rey, CA
11. Mancas C., Dragomir S.: An Optimal Algorithm for Structural Keys CAD in the Elementary Mathematical Data Model. Paper submitted at IASTED SEA2003 International Conference, to be held November 2003 at Marina del Rey, CA