

The (Elementary) Mathematical Data Model Revisited

Type: Research Article

Received: September 08, 2024

Published: September 30, 2024

Citation:

Christian Mancas. "The (Elementary) Mathematical Data Model Revisited". PriMera Scientific Engineering 5.4 (2024): 78-91.

Copyright:

© 2024 Christian Mancas.
This is an open-access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Christian Mancas*

Mathematics and Computer Science Dept., Ovidius University at Constanta, Romania

***Corresponding Author:** Christian Mancas, Mathematics and Computer Science Dept., Ovidius University at Constanta, Romania.

Abstract

This paper presents the current version of our (Elementary) Mathematical Data Model ((E)MDM), which is based on the naïve theory of sets, relations, and functions, as well as on the first-order predicate calculus with equality. Many real-life examples illustrate its 4 types of sets, 4 types of functions, and 76 types of constraints. This rich panoply of constraints is the main strength of this model, guaranteeing that any data value stored in a database is plausible, which is the highest possible level of syntactical data quality. An (E)MDM example scheme is presented and contrasted with some popular family tree software products.

Keywords: (Elementary) Mathematical Data Model; *MatBase*; Naïve theory of sets relations and functions; First order predicate calculus with equality; Database design; Modelware

Introduction

The (Elementary) Mathematical Data Model ((E)MDM) was introduced first in Romania [1-4] and then internationally in [5-7]. Its main advantage is its plethora of constraint types that grew continuously every year, whenever a new real-life example was encountered. The last published paper on (E)MDM [8], considered only 60 explicit constraint types; currently, there are 76.

Data quality is paramount for any software system: if you let a database (db) store unplausible data ("garbage in"), then both information and knowledge computed on that data will be unplausible ("garbage out"). Software systems cannot decide whether a data value is correct or not: for example, most probably, only a handful of people know exactly what was HM Queen Elizabeth II height on her death bed; how could a software application know it? But any software application must accept for db storing only plausible values: for example, for current humans, height must be between 40cm (otherwise chances of survival are almost zero) and 275cm (Robert Wadlaw, the tallest man ever, had 272cm).

Constraints are formalizing business rules. If you do not enforce a single such rule, your db might store unplausible values. Dually, of course, if you add to your software system rules that do not exist in the corresponding subuniverse of discourse, you prevent storing plausible data values in the db. This is why discovering and enforcing all business rules governing the subuniverse you model is crucial.

Classifying constraints in mathematical types is not only beneficial to data modeling but also allows for their algorithmic enforcement through automatically generated software code. Thus, (E)MDM is also a modelware tool, i.e., a member of the family of the 5th generation of programming languages [9].

Database theory is pure applied naïve theory of sets, relations, and functions, as well as of the first-order predicate logic with equality (which formalizes both constraints, as closed clauses, and queries, as open ones). As this is almost entirely taught during K12 studies, we've added (Elementary) to the name of our model.

We introduced (E)MDM as a cornerstone modelling level between the Entity-Relationship Data Models (E-RDM) and Relational Data Model (RDM) schemata [10], for both validating and refining the E-RDM models before translating them into RDM schemata and sets of non-relational constraints that must be enforced by the software applications managing the corresponding dbs.

The following section presents the current version of the (E)MDM. Section 3 presents an example of (E)MDM modeling. Section 4 discusses it against some popular family tree software products. Section 5 abstracts related work. The paper ends with conclusion and a reference list.

The current version of (E)MDM

A(n) (E)MDM db scheme is a quadruple $\mathcal{DKS} = \langle S, \mathcal{M}, C; \mathcal{P} \rangle$, where S is a finite non-empty poset by inclusion, \mathcal{M} is a finite non-empty set of mappings defined on and taking values from the sets of S , C is a finite non-empty set of constraints (i.e., closed Horn clauses of the first-order predicate logic with equality) over sets in S and/or mappings in \mathcal{M} , and \mathcal{P} is a finite set of Datalog $_{\neg}$ programs also over sets in S and mappings in \mathcal{M} . Whenever \mathcal{P} is empty, \mathcal{DKS} is a db scheme, otherwise it is a knowledge base one.

The poset of sets

(S, \subseteq) is a poset of sets, with $S = \Omega \oplus \mathcal{V} \oplus *S \oplus SysS$, where:

- $\Omega = \mathcal{E} \oplus \mathcal{R}$ (the non-empty collection of *object sets*), where:
 - ❖ \mathcal{E} is a non-empty collection of atomic *entity-type sets*, e.g., *PEOPLE*, *COUNTRIES*, *PRODUCTS*.
 - ❖ \mathcal{R} is a collection of *relationship-type sets*, e.g., $NEIGHBOUR_COUNTRIES \subset COUNTRIES \times COUNTRIES$, $STOCKS \subset PRODUCTS \times WAREHOUSES$.
- \mathcal{V} is a non-empty collection of *value sets*, e.g., $R_C = \{“r”, “o”, “y”, “g”, “b”, “i”, “v”\}$, $SEXES = \{“F”, “M”\}$, $[0, 16] \subset NAT(2)$, $ASCII(32) \subset ASCII(n)$, $CURRENCY(8) \subset RAT(10, 2)$, $[1/1/100, Today()] \subset DATETIME$ (with $NAT(n)$ being the subset of naturals of at most n digits, $RAT(n, m)$ the subset of rationals of at most n digits before the decimal point and m after it, $ASCII(n)$ the subset of the freely generated monoid over the ASCII alphabet only including strings of maximum length n , etc.).
- $*S$ is a collection of *computed sets*, e.g., *MALES*, *FEMALES*, *UNPAID_BILLS*, *FREE_COUNTRIES*.
- $SysS$ is a collection of *system sets*, e.g., \emptyset (the empty set), *NULLS* (the distinguished countable set of null values), $BOOLE = \{true, false\}$, $NAT(n)$, $RAT(n, m)$, $ASCII(n)$, $DATETIME$ (the set of date and time values).

In RDM schemata, generally, the object sets are tables, the computed sets are views (queries), the system sets are corresponding db management system (RDBMS) data types, and the value sets are their needed subsets.

The set of mappings

$\mathcal{M} = \mathcal{A} \oplus \mathcal{F} \oplus *M \oplus SysM$, where:

- $\mathcal{A} \subset Hom(S - SysS \oplus \mathcal{V}, \mathcal{V})$ is the non-empty *set of attributes*, e.g., $x : PEOPLE \leftrightarrow NAT(10)$, $FirstName : PEOPLE \rightarrow ASCII(64)$, $BirthDate : PEOPLE \rightarrow [1/1/-6000, Today()]$, $Sex : PEOPLE \rightarrow \{“F”, “M”\}$, $x : COUNTRIES \leftrightarrow NAT(3)$, $CountryName : COUNTRIES \rightarrow ASCII(128)$, $TelPrefix : COUNTRIES \rightarrow NAT(4)$, $Stock : STOCKS \rightarrow [0, 100]$, $Amount : UNPAID_BILLS \rightarrow (0, 100000]$, where \leftrightarrow denotes injections (one-to-one functions) and x is our notation for *object identifiers*, i.e., functions to be implemented as autonumber

surrogate primary keys.

- $\mathcal{F} \subset \text{Hom}(S\text{-SysS} \oplus \mathcal{V}, S\text{-SysS} \oplus \mathcal{V})$ is the non-empty set of *structural functions*, e.g., $\text{BirthPlace} : \text{PEOPLE} \rightarrow \text{CITIES}$, $\text{Capital} : \text{COUNTRIES} \leftrightarrow \text{CITIES}$, $\text{Country} : \text{CITIES} \rightarrow \text{COUNTRIES}$, $\text{Representative} : \text{DISTRICTS} \leftrightarrow \text{PEOPLE}$, $\text{Customer} : \text{UNPAID_BILLS} \rightarrow \text{CUSTOMERS}$.
- $^*\mathcal{M}$ is the set of *computed mappings*, e.g., $\text{BirthCountry} = \text{Country} \circ \text{BirthPlace} : \text{PEOPLE} \rightarrow \text{COUNTRIES}$, $\text{Mother} \bullet \text{Father} : \text{PEOPLE} \rightarrow \text{PEOPLE} \times \text{PEOPLE}$, $\text{Senator1} \bullet \text{Senator2} : \text{STATES} \rightarrow \text{PEOPLE} \times \text{PEOPLE}$, $\text{Age} = \text{Years}(\text{Today}() - \text{BirthDate}) : \text{PEOPLE} \rightarrow [0, 175]$.
- SysM is the set of *system mappings*, e.g., $\mathbf{1}_S$ (the unity function of a set S), $\text{card}(S)$ (the cardinal of a set S), $\text{Im}(f)$, $\text{Prelm}(f)$, $\text{Ker}(f)$ (the image, pre-image, and kernel of a function f), Len , Lft , Rgt , Mid (the length, left, right, and middle parts of a string), etc.

In RDM schemata, the attributes, structural functions, and computed mappings are implemented as table or/and view (query) columns, with structural functions being foreign keys.

The set of constraints

$C = SC \oplus MC \oplus OC \oplus \text{Sys}C$, where:

Set constraints

There are two blocks of set constraints: general and dyadic relation ones.

- $SC = SGC \oplus SDRC$ is the set of *set constraints*, where:
 - ✓ $SGC = SIC \oplus SEC \oplus SDC \oplus SUC \oplus SDSC$ is the set of *general set constraints*, where:
 - ❖ SIC is the set of *inclusion constraints*, e.g., $\text{DRIVERS} \subseteq \text{EMPLOYEES} \subseteq \text{PEOPLE} \subseteq \text{CUSTOMERS}$, $\text{PREEQUISITES} \subseteq \text{COURSES}$.
 - ❖ SEC is the set of *set equality constraints*, e.g., $\text{TAKEOFF_AIRPORTS} = \text{AIRPORTS}$, $\text{LANDING_AIRPORTS} = \text{AIRPORTS}$.
 - ❖ SDC is the set of *disjointness constraints*, e.g., $\text{ELECTED_OFFICIALS} \cap \text{STATE_CONTRACTORS} = \emptyset$, $\text{INCOMPATIBLE_DRUGS} \cap \text{OTHER_DRUGS} = \emptyset$.
 - ❖ SUC is the set of *union constraints*, e.g., $\text{INCOMPATIBLE_DRUGS} = \text{AMINOGLYCOSIDES} \cup \text{CHLORDIAZEPOXIDE} \cup \text{DIAZEPAM} \cup \text{DIGITALIS_GLYCOSIDES} \cup \text{PENTOBARBITAL} \cup \text{PHENYTOIN} \cup \text{SECOBARBITAL} \cup \text{SODIUM_BICARBONATE} \cup \text{THEOPHYLLINE_DERIVATIVES}$.
 - ❖ $SDSC$ is the set of *direct sum constraints*, e.g., $\text{MPS} \subseteq \text{COMMONERS} \oplus \text{LORDS}$, $\text{CONGRESSMEN} = \text{REPRESENTATIVES} \oplus \text{SENATORS}$.

Among these 5 constraint types, only inclusion is fundamental: equalities are double inclusions, and the 3 remaining ones are particular cases of equality.

- ✓ $SDRC = DRRC \oplus DRIRC \oplus DRSC \oplus DRASC \oplus DRIC \oplus DRITC \oplus DREC$.

$DRIEC \oplus DRQC \oplus DRAC \oplus DRCC$ is the set of *dyadic relation constraints*, where:

- ❖ $DRRC$ is the set of *dyadic relation reflexivity constraints*, e.g., HasSameColorAs , HasSameSizeAs , HasSameShapeAs , $\text{IsAtLeastAsComplicatedAs}$, LivesInSameCityAs , IsBloodRelatedTo , WeighsNo-MoreThan .
- ❖ $DRIRC$ is the set of *dyadic relation irreflexivity constraints*, e.g., IsInFrontOf , $\text{OccuredEarlier(Later)Than}$, IsAdjoinTo , IsLargerThan , IsSmallerThan , IsLeftOf , IsRightOf , Prerequisites , $\text{AirportConnections}$, Distances .
- ❖ $DRSC$ is the set of *dyadic relation symmetry constraints*, e.g., HasSameSizeAs , HasSameShapeAs , IsAdjoinTo , HasSameColorAs , LivesInSameCityAs , IsBloodRelatedTo , IsSiblingOf , WeighsNoMoreThan , $\text{IsAtLeastAsComplicatedAs}$, $\text{IsLocatedWithinXmOf}$, IsMarriedTo , WentOnDateWith , $\text{AirportConnections}$.
- ❖ $DRASC$ is the set of *dyadic relation asymmetry constraints*, e.g., IsFatherOf , IsMotherOf , IsChildOf , $\text{IsYounger(Older)Than}$, $\text{OccuredEarlier(Later)Than}$, WeighsMoreThan , IsLargerThan , IsSmallerThan , IsLeftOf , IsRightOf , IsInFrontOf , Prerequisites , Distances .

- ❖ *DRTC* is the set of *dyadic relation transitivity constraints*, e.g., *IsYounger(Older)Than*, *OccurredEarlier(Later)Than*, *Weighs(-No)MoreThan*, *IsLargerThan*, *IsSmallerThan*, *IsBloodRelatedTo*, *IsAtLeastAsComplicatedAs*, *IsInFrontOf*, *HasArrivedSooner(Later)Than*, *IsPoorer(Richer)Than*, *IsAnAncestor(Descendant)Of*, *HasSameShapeAs*, *HasSameSizeAs*, *HasSameColorAs*, *LivesInSameCityAs*, *IsDividing*, *IsLeftOf*, *IsRightOf*, *IsAncestorOf*, *IsDescendantOf*, *AirportConnections*, *Distances*.
- ❖ *DRITC* is the set of *dyadic relation intransitivity constraints*, e.g., *IsMotherOf*, *IsFatherOf*, *IsChildrenOf*, *Prerequisites*.
- ❖ *DRFC* is the set of *dyadic relation Euclidean constraints*, e.g., *IsBloodRelatedTo*, *HasSameShapeAs*, *HasSameSizeAs*, *HasSameColorAs*, *LivesInSameCityAs*, *IsSiblingOf* (in (E)MDM, a Euclidean dyadic relation is both left and right Euclidean).
- ❖ *DRIEC* is the set of *dyadic relation inEuclidean constraints*, e.g., *IsMotherOf*, *IsFatherOf*, *IsChildrenOf*, *IsAncestor* (in (E)MDM, an inEuclidean dyadic relation is neither left, nor right Euclidean).
- ❖ *DRQC* is the set of *dyadic relation equivalence constraints*, e.g., *IsAtLeastAsComplicatedAs*, *HasSameCitizenshipAs*, *HasSame(-Time)Length-As*, *HasSameReligionAs*, *HasSameChildrenAs*, *HasSameEmployerAs*, *Has-Same(Time)LengthAs*, *WeighsNoMoreThan*, *HasSameColorAs*, *HasSame-SizeAs*, *HasSameRaceAs*, *HasSameShapeAs*, *LivesInSameCityAs*, *Graduated-SameSchool(Alumni)As*, *IsBloodRelatedTo*.
- ❖ *DRAC* is the set of *dyadic relation acyclicity constraints*, e.g., *IsFatherOf*, *IsMotherOf*, *IsChildOf*, *IsYounger(Older)Than*, *OccuredEarlier(Later)Than*, *WeighsMoreThan*, *IsLargerThan*, *IsSmallerThan*, *IsLeftOf*, *IsRightOf*, *IsInFrontOf*, *Prerequisites*.
- ❖ *DRCC* is the set of *dyadic relation connectivity constraints*, e.g., *BelongsTo-SameGroupAs*, no matter what group is involved, from the algebraic to social media ones (in (E)MDM, connectivity is weak, i.e., it includes $x \neq y$, for any pair $\langle x, y \rangle$ of connected elements, which is also called by some authors *connex* or *completeness*).

None of these 11 constraint types is fundamental, as dyadic relations are particular cases of homogeneous binary function products (see the corresponding block from $\mathcal{H}BFPC$).

The grand total of set constraint types is 16, with only 1 fundamental.

Mapping constraints

There are five blocks of mapping constraints: general, self-map, function product, homogeneous binary function products, and function diagram cycle ones.

- $MC = MGC \oplus MSMC \oplus MPC \oplus \mathcal{H}BFPC \oplus FDCC$ is the set of *mapping constraints*, where:
 - ✓ $MGC = MTC \oplus MIC \oplus MNPC \oplus MSC \oplus MBC \oplus MDVC$ is the set of *general mapping constraints*, where:
 - ❖ *MTC* is the set of *totality constraints*, e.g., $x : PEOPLE \leftrightarrow NAT(10)$ total, $FirstName : PEOPLE \leftrightarrow ASCII(64)$ total, $x : COUNTRIES \leftrightarrow NAT(3)$ total, $Country-Name : COUNTRIES \rightarrow ASCII(128)$ total, $Country : CITIES \rightarrow COUNTRIES$ total (in (E)MDM, $f : D \rightarrow C$ is total iff $C \cap NULLS = \emptyset$).
 - ❖ *MIC* is the set of *single key (injectivity) constraints*, e.g., $x : PEOPLE \leftrightarrow NAT(10)$, $x : COUNTRIES \leftrightarrow NAT(3)$, $Capital : COUNTRIES \leftrightarrow CITIES$, $Representative : DISTRICTS \leftrightarrow PEOPLE$.
 - ❖ *MNPC* is the set of *non-primeness constraints*, e.g., $Stock : STOCKS \rightarrow [0, 100]$, $Amount : UNPAID_BILLS \rightarrow (0, 100000]$, $Area : COUNTRIES \rightarrow NAT(8)$ (in (E)MDM, $f : D \rightarrow C$ is *non-prime* iff, semantically, it cannot be either one-to-one or a member of a minimally one-to-one function product).
 - ❖ *MSC* is the set of *surjectivity constraints*, e.g., $Edition : VOLUMES \rightarrow EDITIONS$, $County : CITIES \rightarrow COUNTIES$.
 - ❖ *MBC* is the set of *bijectivity constraints*, e.g., $District : REPRESENTATIVES \leftrightarrow DISTRICTS$.
 - ❖ *MDVC* is the set of *default value constraints*, e.g., $Stock : STOCKS \rightarrow [0, 100]$ default = 0.

In total, there are 6 types of general mapping constraints, out of which only one-to-oneness (injectivity), non-primeness, ontoness (surjectivity), and default ones are fundamental: totality is a particular case of existence constraints (i.e., of type $\emptyset \dashv\vdash$ —g, see the mapping product block) and bijectivity derives from injectivity and surjectivity.

- ✓ $MPC = MMIC \oplus MSKC \oplus MEC \oplus MNEC$ is the set of *general mapping product constraints*, where:
 - ❖ $MMIC$ is the set of *concatenated key (minimal injectivity) constraints*, e.g., $StateName \bullet Country : STATES \leftrightarrow ASCII(64) \times COUNTRIES$, $Prerequisite \bullet Course : PREREQUISITES \leftrightarrow COURSES \times COURSES$, $City1 \bullet City2 : DISTANCES \leftrightarrow CITIES \times CITIES$.
 - ❖ $MSKC$ is the set of *subkey (variable geometry key) constraints*, e.g., $Folder \bullet FName$ subkey of $Folder \bullet FName \bullet FExt : FILES \rightarrow FILES \times ASCII(255) \times (ASCII(255) \cup NULLS)$, whenever $FExt(x) \in NULLS$.
 - ❖ MEC is the set of *existence constraints*, e.g., $e-mail \dashv\vdash FName \bullet Phone$, $Fname \bullet Phone \dashv\vdash e-mail$ (in (E)MDM, just like in RDM, $f \dashv\vdash g$ is a shorthand for $(\forall x)(f(x) \notin NULLS \Rightarrow g(x) \notin NULLS)$); the only difference is that both f and g may be computed functions, by composition or/and Cartesian function product).
 - ❖ $MNEC$ is the set of *non-existence constraints*, e.g., $TributaryTo \dashv\vdash Lake \bullet Sea \bullet Ocean \bullet LostInto$, where $TributaryTo : RIVERS \rightarrow RIVERS$, $Lake : RIVERS \rightarrow LAKES$, $Sea : RIVERS \rightarrow SEAS$, $Ocean : RIVERS \rightarrow OCEANS$, and $LostInto : RIVERS \rightarrow GEOGRAPHIC_UNITS$ (in (E)MDM, $f \dashv\vdash g$ is a shorthand for $(\forall x)(f(x) \notin NULLS \Rightarrow g(x) \in NULLS)$).

In total, there are 4 types of general mapping product constraints, all of them being fundamental.

- ✓ $HBFPC = HBFPC \oplus HBFPRC \oplus HBFPNRC \oplus HBFPNIC \oplus HBFPIRC \oplus HBFPPSC \oplus HBFPPNSC \oplus HBFPPASC \oplus HBFPPTC \oplus HBFPPNTC \oplus HBFPPITC \oplus HBFPEEC \oplus HBFPPNEC \oplus HBFPIIEC \oplus HBFPPQC \oplus HBFPPNQC \oplus HBFPPACC$ is the set of *homogeneous binary function product (hbfp) constraints* (i.e., of type $f \bullet g : D \rightarrow C^2$; dyadic relations are particular cases of hbfps, for which $C \cap NULLS = \emptyset$, $f \bullet g$ is minimally one-to-one, and f and g are called *canonical projections*), where:
 - ❖ $HBFPC$ is the set of *hbfp connectivity constraints*, e.g., $Host \bullet Visitor : CHAMPIONSHIP_MATCHES \rightarrow TEAMS \times TEAMS$.
 - ❖ $HBFPRC$ is the set of *hbfp reflexivity constraints*, e.g., $Domain \bullet Codomain : \mathcal{F} \rightarrow S \times S$, as, for any set $S \in S$, there is a corresponding system unity mapping $\mathbf{1}_S : S \rightarrow S$, $\mathbf{1}_S(x) = x$, $\forall x \in S$.
 - ❖ $HBFPNRC$ is the set of *hbfp null-reflexivity constraints*; in (E)MDM, given any dyadic relation constraint type $DRCT$, the null- $DRCT$ corresponding constraint type requires that $DRCT$ be satisfied for any not null-values; e.g., $f \bullet g : D \rightarrow (C \cup NULLS)^2$ is *null-reflexive* iff $(\forall x \in C)(Im(f \bullet g) \supset (x, x) \vee Im(f \bullet g) \supset (x,) \vee Im(f \bullet g) \supset (, x) \vee Im(f \bullet g) \supset (,))$.
 - ❖ $HBFPNIC$ is the set of *hbfp null-identity constraints*, e.g., $BirthPlace \bullet (City \bullet BirthClinic) : PEOPLE \rightarrow CITIES \times CITIES$ (identity ones are useless: why storing in a same db table two columns whose values must be equal on each line?).
 - ❖ $HBFPIRC$ is the set of *hbfp irreflexivity constraints*, e.g., $Host \bullet Visitor : MATCHES \rightarrow TEAMS \times TEAMS$.
 - ❖ $HBFPPSC$ is the set of *hbfp symmetry constraints*, e.g., $TakeOffAirport \bullet LandingAirport : AIRPORT_CONNECTIONS \rightarrow AIRPORTS \times AIRPORTS$.
 - ❖ $HBFPPNSC$ is the set of *hbfp null-symmetry constraints* (i.e., $(\forall x, y \in C)(Im(f \bullet g) \supset (x, y) \Rightarrow Im(f \bullet g) \supset (y, x) \vee Im(f \bullet g) \supset (y,) \vee Im(f \bullet g) \supset (, x) \vee Im(f \bullet g) \supset (,))$).
 - ❖ $HBFPPASC$ is the set of *hbfp asymmetry constraints*, e.g., $Host \bullet Visitor : ELIMINATORY_MATCHES \rightarrow TEAMS \times TEAMS$.
 - ❖ $HBFPPTC$ is the set of *hbfp transitivity constraints*, e.g., $TakeOffAirport \bullet LandingAirport : AIRPORT_CONNECTIONS \rightarrow AIRPORTS \times AIRPORTS$.
 - ❖ $HBFPPNTC$ is the set of *hbfp null-transitivity constraints* (i.e., $(\forall x, y, z \in C)(Im(f \bullet g) \supset \{(x, y), (y, z)\} \Rightarrow Im(f \bullet g) \supset (x, z) \vee Im(f \bullet g) \supset (x,) \vee Im(f \bullet g) \supset (, z) \vee Im(f \bullet g) \supset (,))$).
 - ❖ $HBFPIITC$ is the set of *hbfp intransitivity constraints*, e.g., $Host \bullet Visitor : ELIMINATORY_MATCHES \rightarrow TEAMS \times TEAMS$.
 - ❖ $HBFPEEC$ is the set of *hbfp Euclidean constraints*, e.g., $Host \bullet Visitor : CHAMPIONSHIP_MATCHES \rightarrow TEAMS \times TEAMS$ (in (E)MDM, Euclidean means both left and right Euclidean).
 - ❖ $HBFPPNEC$ is the set of *hbfp null-Euclidean constraints* (i.e., $(\forall x, y, z \in C)(Im(f \bullet g) \supset \{(x, y), (y, z)\} \Rightarrow Im(f \bullet g) \supset (y, z) \vee Im(f \bullet g) \supset (y,) \vee Im(f \bullet g) \supset (, z) \vee Im(f \bullet g) \supset (,) \wedge (Im(f \bullet g) \supset \{(x, y), (z, y)\} \Rightarrow Im(f \bullet g) \supset (x, z) \vee Im(f \bullet g) \supset (x,) \vee Im(f \bullet g) \supset (, z) \vee Im(f \bullet g) \supset (,))$).
 - ❖ $HBFPIIEC$ is the set of *hbfp inEuclidean constraints*, e.g., $Host \bullet Visitor : ELIMINATORY_MATCHES \rightarrow TEAMS \times TEAMS$ (in (E)MDM, inEuclidean means neither left, nor right Euclidean).

- ❖ $\mathcal{H}BFPQT$ is the set of *hbf* equivalence constraints, e.g., $TakeOffAirport \bullet LandingAirport : AIRPORT_CONNECTIONS \rightarrow AIRPORTS \times AIRPORTS$.
- ❖ $\mathcal{H}BFPNQC$ is the set of *hbf* null-equivalence constraints (i.e., $f \bullet g$ is null-equivalent iff it is null-reflexive and null-Euclidean or reflexive and null-Euclidean or null-reflexive and Euclidean).
- ❖ $\mathcal{H}BFPACC$ is the set of *hbf* acyclicity constraints, e.g., $Host \bullet Visitor : ELIMINATORY_MATCHES \rightarrow TEAMS \times TEAMS$.

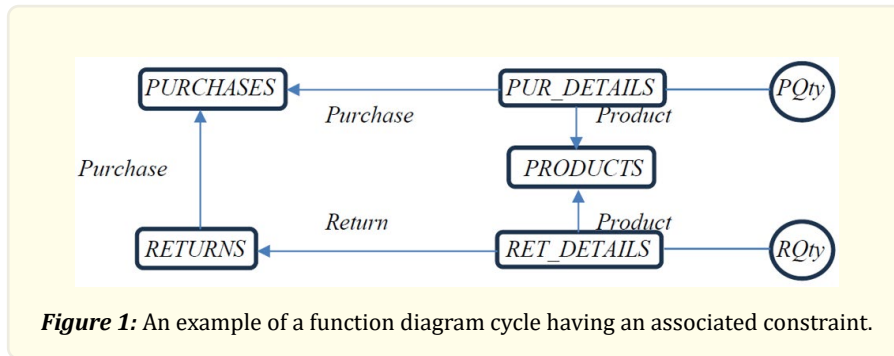
In total, there are 17 *hbf* constraint types, out of which only connectivity, reflexivity, null-identity, irreflexivity, symmetry, asymmetry, transitivity, intransitivity, Euclideanity, inEuclideanity, and acyclicity are fundamental: equivalence is derived from reflexivity and Euclideanity, and the other 5 null-type ones are trivially derived from their not-null counterparts.

- ✓ $MSMC = SMRC \oplus SMNRC \oplus SMIRC \oplus SMSC \oplus SMNSC \oplus SMASC \oplus SMIC \oplus SMNISC \oplus SMAIC \oplus SMRSC \oplus SMNRSC \oplus SMAC \oplus SMQC \oplus SMNQC$ is the set of *self-map* constraints, where:
 - ❖ $SMRC$ is the set of *self-map reflexivity* constraints, e.g., $Country \circ State \circ Capital$ total, reflex, $State \circ Capital$ total.
 - ❖ $SMNRC$ is the set of *self-map null-reflexivity* constraints, e.g., $Country \circ State \circ Capital$ reflex, $State \circ Capital$ (i.e., Capital might take null values as well).
 - ❖ $SMIRC$ is the set of *self-map irreflexivity* constraints, e.g., $Folder : FILES \rightarrow FILES$, $ReportTo : EMPLOYEES \rightarrow EMPLOYEES$, $Mother : PEOPLE \rightarrow PEOPLE$, $Father : PEOPLE \rightarrow PEOPLE$.
 - ❖ $SMSC$ is the set of *self-map symmetry* constraints, e.g., $Spouse : MARRIED_PEOPLE \rightarrow MARRIED_PEOPLE$ total
 - ❖ $SMNSC$ is the set of *self-map null-symmetry* constraints, e.g., $Spouse : PEOPLE \rightarrow PEOPLE$ (i.e., *Spouse* might take null values as well).
 - ❖ $SMASC$ is the set of *self-map asymmetry* constraints, e.g., $Folder : FILES \rightarrow FILES$, $ReportTo : EMPLOYEES \rightarrow EMPLOYEES$, $Mother : PEOPLE \rightarrow PEOPLE$, $Father : PEOPLE \rightarrow PEOPLE$.
 - ❖ $SMIC$ is the set of *self-map idempotency* constraints, e.g., $LocalRepres : CITIZENS \rightarrow CITIZENS$ total.
 - ❖ $SMNIC$ is the set of *self-map null-idempotency* constraints, e.g., $ReplacementPart : PART_TYPES \rightarrow PART_TYPES$ (i.e., *ReplacementPart* might take null values as well).
 - ❖ $SMAIC$ is the set of *self-map anti-idempotency* constraints, e.g., $Folder : FILES \rightarrow FILES$, $ReportTo : EMPLOYEES \rightarrow EMPLOYEES$, $Mother : PEOPLE \rightarrow PEOPLE$.
 - ❖ $SMRSC$ is the set of *representative system mapping* constraints, e.g., $Representative : USCITIZENS \rightarrow REPRESENTATIVES$, $Representative = RepresentedBy \circ District$, where $District$ is the canonical surjection $District : USCITIZENS \rightarrow DISTRICTS$ and $RepresentedBy : DISTRICTS \leftrightarrow REPRESENTATIVES \subseteq USCITIZENS$ is the canonical identification mapping of the corresponding representative system.
 - ❖ $SMNCSC$ is the set of *null-representative system mapping* constraints, e.g., $ReportsTo : EMPLOYEES \rightarrow EMPLOYEES$ (i.e., *ReportsTo* might take null values as well, as, generally, the roots of hierarchical organizations do not report to anybody).
 - ❖ $SMQC$ is the set of *self-map equivalence* constraints, e.g., $State \circ Capital$ total.
 - ❖ $SMNQC$ is the set of *self-map null-equivalence* constraints, e.g., $State \circ Capital$ (i.e., *Capital* might take null values as well).
 - ❖ $SMAC$ is the set of *self-map acyclicity* constraints, e.g., $Folder : FILES \rightarrow FILES$, $ReportTo : EMPLOYEES \rightarrow EMPLOYEES$, $Mother : PEOPLE \rightarrow PEOPLE$, $Father : PEOPLE \rightarrow PEOPLE$.

In total, there are 14 types of *self-map* constraints, none of which being fundamental, as *self-maps* are particular cases of dyadic relations (where $\mathbf{1}_D$ and $f : D \rightarrow D$ are the canonical projections).

- ✓ $FDCC = FDEC \oplus FDNCC \oplus FDACC \oplus FDLCC \oplus FDLNCC \oplus FDLACC \oplus FDGCC \oplus FDLSC \oplus FDLNSC \oplus FDLASC \oplus FDLIC \oplus FDLNIC \oplus FDLAIC \oplus FDLQC \oplus FDLNQC \oplus FDLACC \oplus FDLRSC \oplus FDLNCRSC$ is the set of *function diagram cycle* constraints, where:
 - ❖ $FDEC$ is the set of *function diagram commutativity (equality)* constraints, e.g., $PC \circ LogicDrive = Host \circ DBMS \circ DB$ ("any file which is managed by a DBMS should belong to a logic drive of the PC hosting that DBMS").

- ❖ \mathcal{FDNCC} is the set of *function diagram null-commutativity constraints*, e.g., $RContinent = MContinent \circ Range \circ Subrange \circ Group \circ Mountain$ (“any river that springs from a mountain belongs to the same continent as that mountain”; both *Mountain* and *Group* might take null values as well).
- ❖ \mathcal{FDACC} is the set of *function diagram anti-commutativity (inequality) constraints*, e.g., $(\forall x \in NEIGHBOR_COUNTRIES)(FrontierColor(Country(x)) \neq FrontierColor(Neighbor(x)))$.
- ❖ \mathcal{FDLCC} is the set of *function diagram local commutativity (reflexivity) constraints*, e.g., $ConstrRelation \circ PrimaryKey = \mathbf{1}_{RELATIONS}$ (“the primary key of any relation is a constraint of that relation”).
- ❖ \mathcal{FDLNCC} is the set of *function diagram local null-commutativity constraints*, e.g., $Country \circ State \circ Capital = \mathbf{1}_{COUNTRIES}$ (“the capital of a country must be a city of that country”; *Capital* might take null values as well).
- ❖ \mathcal{FDLACC} is the set of *function diagram local anti-commutativity (irreflexivity) constraints*, e.g., $Spouse \circ Mother, Spouse \circ Father, Mother \circ Spouse, Father \circ Spouse$.
- ❖ \mathcal{FDLSC} is the set of *function diagram local symmetry constraints*, e.g., $Spouse \circ \mathbf{1}_{PEOPLE}$ total.
- ❖ \mathcal{FDLNSC} is the set of *function diagram local null-symmetry constraints*, e.g., $Spouse \circ \mathbf{1}_{PEOPLE}$ (i.e., *Spouse* might take null values as well).
- ❖ \mathcal{FDLASC} is the set of *function diagram local asymmetry constraints*, e.g., $Spouse \circ Mother, Spouse \circ Father, Mother \circ Spouse, Father \circ Spouse$.
- ❖ \mathcal{FDLIC} is the set of *function diagram local idempotency constraints*, e.g., $Country \circ State \circ Capital$ total.
- ❖ \mathcal{FDLNIC} is the set of *function diagram local null-idempotency constraints*, e.g., $Country \circ State \circ Capital$ (i.e., *Capital* might take null values as well).
- ❖ \mathcal{FDLAIC} is the set of *function diagram local anti-idempotency constraints*, e.g., $Spouse \circ Mother, Spouse \circ Father, Mother \circ Spouse, Father \circ Spouse$.
- ❖ \mathcal{FDLQC} is the set of *function diagram local equivalence constraints*, e.g., $State \circ Capital$ total.
- ❖ \mathcal{FDLNQC} is the set of *function diagram local null-equivalence constraints*, e.g., $State \circ Capital$ (i.e., *Capital* might take null values as well).
- ❖ \mathcal{FDLACC} is the set of *function diagram local acyclicity constraints*, e.g., $Spouse \circ Mother, Spouse \circ Father, Mother \circ Spouse, Father \circ Spouse$.
- ❖ \mathcal{FDLRSC} is the set of *function diagram local representative system mapping constraints*, e.g., $currentMP \circ EDistrict : UK_VOTERS \rightarrow UK_VOTERS$, where $currentMP : ELECTORAL_DISTRICTS \rightarrow UK_VOTERS$ total and $EDistrict : UK_VOTERS \rightarrow ELECTORAL_DISTRICTS$ total.
- ❖ $\mathcal{FDLNRSC}$ is the set of *function diagram local null-representative system mapping constraints*, e.g., $RepresentedBy = currentMP \circ EDistrict : UK_VOTERS \rightarrow UK_VOTERS$, where $currentMP : ELECTORAL_DISTRICTS \rightarrow UK_VOTERS$ and $EDistrict : UK_VOTERS \rightarrow ELECTORAL_DISTRICTS$ (i.e., both *currentMP* and *EDistrict* might take null values as well).
- ❖ \mathcal{FDGCC} is the set of *function diagram general commutativity constraints*, e.g., let us consider the function diagram from Figure 1 and its associated general commutativity constraint $retPurCnstr: (\forall x \in PUR_DETAILS)(\forall y \in RET_DETAILS)(Purchase(x) = Purchase(Return(y)) \wedge Product(x) = Product(y) \Rightarrow Rqty(y) \leq Pqty(x))$ (“You cannot return more products than you have purchased”).



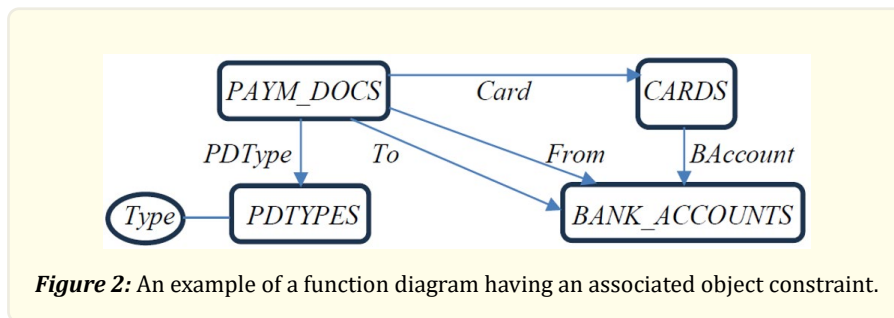
In total, there are 18 types of function diagram cycle constraints, out of which only the commutativity, general commutativity, and anti-commutativity are fundamental: the local-type ones are particular cases of self-maps, namely computed self-maps through function composition.

The grand total for the mapping constraint set is 59 types, with 22 ones fundamental.

Object constraints

In (E)MDM, any other explicit constraint (i.e., which is not of any of the above 75 types) is called an *object constraint* and is formalized by a closed Horn clause. This type is fundamental: in fact, any other constraint type is ultimately formalized by a closed Horn clause.

- *OC* is the set of object constraints, e.g., let us consider the function diagram from Figure 2 (where $Im(Type) = \{“cash”, “card”, “wire”\}$), functions, and associated object (*PD1*) and non-existence (*PD2* and *PD3*) constraints:
 - PD1*: $(\forall x \in PAYM_DOCS)(PDType(Type(x)) = “cash” \Rightarrow To(x) \in NULLS \wedge From(x) \in NULLS \wedge Card(x) \in NULLS)$ (“Whenever *Type* is “cash”, *To*, *From*, and *Card* must be null”).
 - PD2*: $To \bullet Card \neg \vdash \neg From$ (“If *Type* is “card”, *From* must be null, whereas *To* and *Card* must not be null”).
 - PD3*: $To \bullet From \neg \vdash \neg Card$ (“If *Type* is “wire”, *Card* must be null and *To* and *From* must not be null”).



In total, (E)MDM provides 76 types of explicit constraints, out of which 24 are fundamental.

The 6 types of relational constraints provided by RDBMSes (range ((co)domain), not null, keys, referential integrity (foreign keys), tuple (check), and default value) should be enforced through their engines. All other constraint types (i.e. non-relational) should be enforced by the software applications managing the corresponding dbs, through their event-driven methods and embedded SQL.

System constraints

Apart from the explicit ones, any db scheme also includes implicit *system constraints* (all being transparent to users). Their types and numbers vary in function of the DBMS versions implementing them. The following are among the most common ones:

- $SysC$ is the set of *system constraints*, e.g.,
 - $\emptyset \subseteq S, \forall S \in S.$
 - $card(\emptyset) = 0; card(S) \geq 0, \forall S \in S.$
 - $\emptyset \cup S = S, \emptyset \oplus S = S, \forall S \in S.$
 - $\emptyset \cap S = \emptyset, S \cap S = S, S \cup S = S, \forall S \in S.$
 - $T = S \Leftrightarrow T \subseteq S \wedge S \subseteq T, \forall S, T \in S.$
 - $U = S \oplus T \Leftrightarrow S \cap T = \emptyset \wedge S \cup T = U, \forall S, T, U \in S.$
 - $NAT(n) \subseteq INT(n) \subseteq CURRENCY(n) \subseteq RAT(n + m, m),$ for any naturals n and $m > 1.$
 - $NAT(n) \subset NAT(m), INT(n) \subset INT(m), \dots, ASCII(n) \subset ASCII(m),$ for any naturals $n < m.$
 - $\forall R \in \mathcal{R}, R = (f1 \rightarrow C, \dots, fn \rightarrow C), f1 \text{ total} \wedge \dots \wedge fn \text{ total} \wedge f1 \bullet \dots \bullet fn \text{ injective.}$

An (E)MDM scheme example

Figure 3 presents, as an example, the (E)MDM scheme for a simple subuniverse of interest: people and marriages (in legislations in which polygamy is not allowed).

Discussion

For example, you can test a week for free all following popular tree family software products and, if you use your imagination, you will be amazed by how poor is their concern for your data quality. To exemplify with only a few commonsense tests, please note that, with the products Legacy Family Tree (LFT) [11], RootsMagic (RM) [12], GenoPro (GP) [13], Gramps (G) [14], Family Historian (FH) [15], Family Tree Heritage (FTH) [16], and Ancestral Quest (AQ) [17] you can store in your db values according to which any person was:

- born after his/her death (LFT, RM, GP, G, FH, FTH, AQ).
- baptized before birth (LFT, RM, G, FTH, AQ).
- buried before death (RM, G, FTH, AQ).
- having a male as mother (FH).
- born before his/her parents' birth or hundreds of years after their death (GP, G, FH, FTH, AQ).
- having a same person as both his/her mother and father (G).
- married with a person of same sex and had together a child dead before their birth (G).

It is true that, for some values of such totally unplausible data, FH, FTH, and AQ (a clone of FTH) at least display a warning reading that they might be wrong, but if you ignore it you may save them in the db.

Obviously, thanks to its constraints, the (E)MDM scheme presented in Figure 3 (correspondingly extended with baptism and burial dates, as well as corresponding constraints) would reject storing any such unplausible data.

Related work

Besides its system constraints, (E)MDM also includes dozens of meta-constraints for guaranteeing minimality of constraint sets and assisting its users in also guaranteeing their coherence [8]. For example, as acyclicity implies asymmetry, which implies irreflexivity, both asymmetry and irreflexivity are redundant for acyclic graphs (be them dyadic relations or self-maps or homogeneous binary function products), while acyclicity and reflexivity or/and symmetry are incoherent.

PEOPLE (The set of world persons of interest.)

$x \leftrightarrow \text{AUTON}(38)$, total

$FName \rightarrow \text{ASCII}(32)$, total (First names are mandatory.)

$LName \rightarrow \text{ASCII}(128)$, total (Last name is mandatory.)

$Sex \rightarrow \{ 'F', 'M' \}$, total (Sex is mandatory.)

$BirthDate \rightarrow [1/1/1900, \text{Today}()$], total (Birth date is mandatory.)

$Died \rightarrow [1/1/1900, \text{Today}()$]

$SSN \rightarrow \text{NAT}(9)$, total (SSNs are mandatory.)

$Mother : PEOPLE \rightarrow PEOPLE$, acyclic

$Father : PEOPLE \rightarrow PEOPLE$, acyclic

$BirthPlace : PEOPLE \rightarrow CITIES$

$HomeTown : PEOPLE \rightarrow CITIES$, total (Hometowns are mandatory.)

* $HomeCountry = Country \circ State \circ HomeTown$

peopleKey: $SSN \bullet *HomeCountry$ key (There may not be two persons of a same country having same SSNs.)

$C_0: (\forall x \in PEOPLE)(Died(x) \neq \text{NULLS} \Rightarrow BirthDate(x) \leq Died(x))$ (Nobody may die before being born.)

$C_5: (\forall x \in PERSONS)(BirthDate(x) + 160 * 365 \geq Died(x) \geq BirthDate(x))$ (nobody may live more than 160 years)

$C_6: (\forall x \in PERSONS)(Sex(Mother(x)) = 'F')$ (only women may be mothers)

$C_7: (\forall x \in PERSONS)(Sex(Father(x)) = 'M')$ (only men may be fathers)

Figure 3: An example of an (E)MDM scheme.

$C_8: (\forall x \in PERSONS)(BirthDate(x) \leq Died(Mother(x)) \wedge BirthDate(x) + 5 * 365 \leq BirthDate(Mother(x)) \leq BirthDate(x) + 75 * 365)$ (i.e., no mother may give birth after her death, before being 5 years old, or after being 75 years old)

$C_9: (\forall x \in PERSONS)(BirthDate(x) \leq Died(Father(x)) + 10 * 30 \wedge BirthDate(x) + 9 * 365 \leq BirthDate(Father(x)) \leq BirthDate(x) + 100 * 365)$ (i.e., no father may have a child after 10 months from his death, before being 9 years old, or after being 100 years old)

MARRIAGES (The set of marriages of interest.)

$x \leftrightarrow \text{AUTON}(38)$, total

$MarriageDate \rightarrow [1/1/1900, \text{Today}()$], total (Marriage date is mandatory)

$DivorceDate \rightarrow [1/1/1900, \text{Today}()$]

$Husband : MARRIAGES \rightarrow PEOPLE$, total

$Wife : MARRIAGES \rightarrow PEOPLE$, total

$C_1: (\forall x \in MARRIAGES)(Sex(Wife(x)) = 'F' \wedge Sex(Husband(x)) = 'M')$
(Both wives and husbands should have corresponding sexes.)

$C_2: (\forall x \in MARRIAGES)(DivorceDate(x) \neq \text{NULLS} \Rightarrow MarriageDate(x) < DivorceDate(x))$ (Nobody can divorce before being married.)

$C_3: (\forall x \in MARRIAGES)(MarriageDate(x) \geq BirthDate(Wife(x)) + 16 * 365 \wedge MarriageDate(x) \geq BirthDate(Husband(x)) + 16 * 365 \wedge (Died(Wife(x)) \in \text{NULLS} \vee MarriageDate(x) < Died(Wife(x))) \wedge (Died(Husband(x)) \in \text{NULLS} \vee MarriageDate(x) < Died(Husband(x))))$ (Nobody can get married before being 16 years old or after dying.)

Figure 3: (Continued).

$C_4: (\forall x \in \text{MARRIAGES})(\text{DivorceDate}(x) \notin \text{NULLS} \Rightarrow (\text{Died}(\text{Wife}(x)) \notin \text{NULLS} \Rightarrow \text{DivorceDate}(x) < \text{Died}(\text{Wife}(x)))) \wedge (\text{Died}(\text{Husband}(x)) \notin \text{NULLS} \Rightarrow \text{DivorceDate}(x) < \text{Died}(\text{Husband}(x))))$ (Nobody may divorce after dying.)

C_{10} : Husband • MarriageDate key (nobody may marry same day more than once)

C_{11} : Wife • MarriageDate key (nobody may marry same day more than once)

C_{12} : Husband • DivorceDate key (nobody may divorce same day more than once)

C_{13} : Wife • DivorceDate key (nobody may divorce same day more than once)

$C_{14}: (\forall x, y \in \text{MARRIAGES}, x \neq y)(\text{Husband}(x) = \text{Husband}(y) \Rightarrow \text{DivorceDate}(x) \notin \text{NULLS} \wedge \text{MarriageDate}(y) > \text{DivorceDate}(x) \vee \text{DivorceDate}(y) \notin \text{NULLS} \wedge \text{MarriageDate}(x) > \text{DivorceDate}(y))$ (polygamy is not allowed: one can remarry only after divorcing)

$C_{15}: (\forall x, y \in \text{MARRIAGES}, x \neq y)(\text{Wife}(x) = \text{Wife}(y) \Rightarrow \text{DivorceDate}(x) \notin \text{NULLS} \wedge \text{MarriageDate}(y) > \text{DivorceDate}(x) \vee \text{DivorceDate}(y) \notin \text{NULLS} \wedge \text{MarriageDate}(x) > \text{DivorceDate}(y))$ (polygamy is not allowed: one can remarry only after divorcing)

$C_{16}: (\forall x \in \text{MARRIAGES})(\text{Died}(\text{Husband}(x)) > \text{MarriageDate}(x) \vee \text{Died}(\text{Husband}(x)) \notin \text{NULLS}) \wedge \text{MarriageDate}(x) > 16 * 365 + \text{BirthDate}(\text{Husband}(x))$ (nobody can marry before being 16 or after his death)

$C_{17}: (\forall x \in \text{MARRIAGES})(\text{Died}(\text{Wife}(x)) > \text{MarriageDate}(x) \vee \text{Died}(\text{Wife}(x)) \notin \text{NULLS}) \wedge \text{MarriageDate}(x) > 16 * 365 + \text{BirthDate}(\text{Wife}(x))$ (nobody can marry before being 16 or after her death)

$C_{18}: (\forall x \in \text{MARRIAGES})(\text{Died}(\text{Husband}(x)) > \text{DivorceDate}(x) \vee \text{Died}(\text{Husband}(x)) \notin \text{NULLS})$ (nobody can divorce after his death)

$C_{19}: (\forall x \in \text{MARRIAGES})(\text{Died}(\text{Wife}(x)) > \text{DivorceDate}(x) \vee \text{Died}(\text{Wife}(x)) \notin \text{NULLS})$ (nobody can divorce after her death)

Figure 3: (Continued).

We discussed the extraordinary importance of db constraints for guaranteeing data quality in [18, 19].

We used the (E)MDM to model the RDM [6] and the first order predicate logic with equality [20].

MatBase is our intelligent data and knowledge management system prototype, based on both (E)MDM, E-RDM, and RDM [21-23].

We published several algorithms for detecting some of the (E)MDM constraint types [24-27], as well as for enforcing them in *MatBase* [26-32]. Papers [33, 34] deal with the implementation and usage, respectively, of the *MatBase*'s Datalog \rightarrow subsystem.

We proved that the expressive powers of E-RDM [10, 35, 36] and the Functional Data Model (FDM) [37] (which was the data model that inspired (E)MDM) are equivalent [38].

Apart from the FDM, other data modeling approaches are related to (E)MDM: categorical [39], graph [40-43], incomplete dbs [44], probabilistic [45], possibilistic [46], and constraint dbs [47].

Finally, the most closely related approaches to non-relational constraint enforcement are based on business rules management (BRM) [48, 49] and their corresponding implemented systems (BRMS) and process managers (BPM), like the IBM Operational Decision Manager [50], IBM Business Process Manager [51], Red Hat Decision Manager [52], Agiloft Custom Workflow/BPM [53], etc. They are generally based on XML (but also on the Z notation, Business Process Execution Language, Business Process Modeling Notation, Decision Model and Notation, or the Semantics of Business Vocabulary and Business Rules), which is the only other field of endeavor

trying to systematically deal with business rules, even if informally, not at the db design level but at the software application one, and without providing automatic code generation.

From this perspective, (E)MDM is also a BRM but a formal one, and *MatBase* is also a BRMS but an automatically code generating one.

Data quality may be further improved only semantically, for example by checking values against public and/or private trusted dbs, like Geographic Data [54], Top Companies Database [55], Yellow Pages Scraper [56], Amazon Product Database [57].

Conclusion

We revisited (E)MDM, including all its 76 current constraint types, which are its main strength in guaranteeing db data plausibility, the highest possible level of syntactical data quality.

We provided lot of real-life examples for all (E)MDM set, mapping, and constraint types.

We also provided clues on implementing (E)MDM schemata in RDBMSes and software applications managing their dbs.

We presented an (E)MDM scheme for a family subuniverse and compared it with some popular family tree software products, proving that it would reject any of the absurd unplausible data value that they accept for storing in their dbs.

We provided a comprehensive batch of related work, illustrated with a rich corresponding reference list, even if it contains only a few seminal items for any other related data modeling approach or DBMS.

Conflict of Interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Funding

This research received no external funding.

Acknowledgments

This research was not sponsored by anybody and nobody other than its author contributed to it.

References

1. Mancas C. "A first introduction to a data model based on the semi-naïve theory of sets, relations, and functions". In Proc. 5th Conf. on Informatics (INFO-IASI'85), A.I. Cuza Univ., Iasi, Romania (1985): 314-320. (in Romanian).
2. Mancas C. "A Deeper Insight into the Mathematical Data Model". In Proc. 13th Int. Seminar on Database Management Systems (ISDBMS'90), ICI Bucharest, Romania (1990): 122-134.
3. Mancas C. "Conceptual data modelling". Ph. D. Dissertation (in Romanian), Politehnica Univ., Bucharest, Romania (1997).
4. Mancas C. "Normalization versus correctness of conceptual data modelling". Compulsory object constraints in the Elementary Mathematical Data Model too. In Proc. 1st Conf. on Informatics Theory and Technologies (CITTI'2000), Ovidius Univ., Constanta, Romania (2000): 207-216 (in Romanian).
5. Mancas C. "On Modeling Closed Entity-Relationship Diagrams in an Elementary Mathematical Data Model". In Proc. 6th East-European Conf. on Advances in Databases and Inf. Syst. (ADBIS 2002), Slovak Univ. of Techn., Bratislava, Slovakia 2 (2002): 165-174.
6. Mancas C. "On Modeling the Relational Domain-Key Normal Form Using an Elementary Mathematical Data Model". In Proc. 6th IASTED Int. Conf. on Soft. Eng. and App. (SEA 2002), ACTA Press, Calgary, Canada (2002): 767-772.
7. Mancas C. "On Knowledge Representation Using an Elementary Mathematical Data Model". In Proc. 1st IASTED Int. Conf. on Inf.

- and Knowl. Sharing (IKS 2002), ACTA Press, Calgary, Canada (2002): 206-211.
8. Mancas C. "MatBase Constraint Sets Coherence and Minimality Enforcement Algorithms". In: Benczur, A., Thalheim, B., Horvath, T. (eds.), Proc. 22nd ADBIS Conf. on Advances in DB and Inf. Syst., LNCS, Springer, Cham, Switzerland 11019 (2018): 263-277.
 9. Mancas C. "On Modelware as the 5th Generation of Programming Languages". Acta Scientific Computer Sciences 2.9 (2020): 24-26.
 10. Mancas C. "Conceptual Data Modeling and Database Design: A Completely Algorithmic Approach". Volume I: The Shortest Advisable Path. Apple Academic Press / CRC Press (Taylor & Francis Group), Palm Bay, FL (2015).
 11. Millennia. Legacy Family Tree (2024). <https://legacyfamilytree.com/>,
 12. RootsMagic. RootsMagic Essentials 10 (2024). <https://www.rootsmagic.com/try/>.
 13. GenoPro Ltd. GenoPro 2020 (2024). <https://genopro.com/>.
 14. Gramps. Gramps-Generational Research Software (2024). <https://gramps-project.org/blog/>.
 15. Calico Pie Ltd. Family Historian 7 (2024). <https://www.family-historian.co.uk/>.
 16. Individual Software. Family Tree Heritage (2023). <https://www.individualsoftware.com/product/family-tree-heritage-platinum-9-2/>.
 17. Incline Software. Ancestral Quest (2024). <http://www.ancestry.com/index.htm>.
 18. Mancas C. "On the paramount importance of database constraints". J. Inf. Tech. & Soft. Eng. Henderson, NV 5.3 (2015): 1-4.
 19. Mancas C. "On the Preeminence of Data Quality". Acta Scientific Computer Sciences 3.12 (2021b): 26-29.
 20. Mancas C, Dragomir S and Crasovschi L. "On Modeling First Order Predicate Calculus Using the Elementary Mathematical Data Model in MatBase DBMS". In Proc. 25th IASTED Int. Conf. on Databases and App. (DBA 2003), ACTA Press, Calgary, Canada (2003): 1197-1202.
 21. Mancas C. "MatBase-a Tool for Transparent Programming while Modeling Data at Conceptual Levels". In: Proc. 5th Int. Conf. on Comp. Sci. & Inf. Techn. (CSITEC 2019), AIRCC Pub. Corp. Chennai, India (2019): 15-27.
 22. Mancas C. "MatBase Metadata Catalog Management". Acta Scientific Computer Sciences 2.4 (2020): 25-29.
 23. Mancas C and Dorobantu V. "On enforcing relational constraints in MatBase". London Journal of Research in Computer Science and Technology 17.1 (2017): 39-45.
 24. Mancas C. "Algorithms for Key Discovery Assistance". In: Repa, V., Bruckner, T. (eds). BIR 2016, Lecture Notes in Business Information Processing. Springer, Cham, Switzerland 261 (2016): 322-338.
 25. Mancas C. "MatBase E-RD Cycles Associated Non-Relational Constraints Discovery Assistance Algorithm". In: Arai, K., Bhatia, R., Kapoor, S. (eds.), Intelligent Computing, Proc. 2019 Computing Conference, AISC Series. Springer, Cham, Switzerland 997.1 (2019): 390-409.
 26. Mancas C. "On Detecting and Enforcing the Non-Relational Constraints Associated to Dyadic Relations in MatBase". J. of Electronic & Inf. Syst 2.2 (2020):1-8.
 27. Mancas C. "On Variable Geometry Database Keys and their Subkeys". WJAETS 6.2 (2022): 71-80.
 28. Mancas C. "Matbase Autofunction Non-relational Constraints Enforcement Algorithms". IJCSIT 11.5 (2019): 63-76.
 29. Mancas C. "On enforcing dyadic relationship constraints in MatBase". WJAETS 09.02 (2023): 298-311.
 30. Mancas C. "On enforcing dyadic-type self-map constraints in MatBase". IJFETR 05.01 (2023): 014-026.
 31. Mancas C. "On Enforcing Dyadic-Type Homogeneous Binary Function Product Constraints in MatBase". JCSR 6.1 (2024): 31-42.
 32. Mancas C and Mancas DC. "On enforcing existence and non-existence constraints in MatBase". PriMera Scientific Engineering 4.6 (2024): 04-12.
 33. Mancas C and Dragomir S. "MatBase Datalog Subsystem Metacatalog Conceptual Design". In Proc. 8th IASTED Int. Conf. on Soft. Eng. and App. (SEA 2004), ACTA Press, Calgary, Canada (2004): 34-41.
 34. Mancas C. "On MatBase's algorithm for preventing cycles in binary Cartesian function products". WJAETS 7.1 (2022): 23-37.
 35. Chen PP. "The entity-relationship model: Toward a unified view of data". ACM TODS 1.1 (1976):9-36.
 36. Thalheim B. "Entity-Relationship Modeling: Foundations of Database Technology". Springer-Verlag, Berlin (2000).

37. Gray PMD., et al. "The Functional Approach to Data Management". Springer, Berlin Heidelberg, New York (2004).
38. Mancas C and Dragomir S. "On the Equivalence between Entity-Relationship and Functional Data Modeling". In Proc. 7th IASTED Int. Conf. on Soft. Eng. and App. (SEA 2003), ACTA Press, Calgary, Canada (2003): 335-340.
39. Schultz P and Wisnesky R. Algebraic Data Integration. Expanded and corrected version of the paper published in J. Functional Programming, Cambridge Univ. Press 27 (2017): E24.
40. Gosnell D and Broecheler M. "The Practitioner's Guide to Graph Data: Applying Graph Thinking and Graph Technologies to Solve Complex Problems". O'Reilly Media, Inc., CA (2020).
41. Neo4j. Get started with Neo4j (2024). <https://neo4j.com/docs/getting-started/>
42. Microsoft. Azure Cosmos DB (2024). <https://azure.microsoft.com/en-us/products/cosmos-db>
43. AWS. Amazon Neptune (2024). <https://aws.amazon.com/neptune/>
44. Green T and Tannen V. "Models for incomplete and probabilistic information". IEEE Data Engineering Bulletin 29.1 (2006): 17-24.
45. Kenig B and Suciu D. "Integrity Constraints Revisited: From Exact to Approximate Implication". In Proc. 23rd Intl. Conf. on DB Theory (ICDT 2020) 18 (2020).
46. Roblot TK and Link S. "Possibilistic Cardinality Constraints and Functional Dependencies". In: Comyn-Wattiau, I. et al. (eds.) Conceptual Modeling. ER 2016. LNCS, Springer, Cham 9974 (2016): 133-148.
47. NASA. EOSCUBE: A Constraint Database System for High-Level Specification and Efficient Generation of EOSDIS Products. Phase 1; Proof-of-Concept. NASA, Washington D.C (2019).
48. Morgan T. "Business Rules and Information Systems: Aligning IT with Business Goals". Addison-Wesley Professional, Boston, MA (2002).
49. von Halle B and Goldberg L. "The Business Rule Revolution. Running Businesses the Right Way". Happy About, Cupertino, CA (2006).
50. IBM Corp. Introducing Operational Decision Manager (2024). <https://www.ibm.com/docs/en/odm/8.12.0?topic=manager-introducing-operational-decision>.
51. Dyer L., et al. (Scaling BPM Adoption from Project to Program with IBM Business Process Manager 2012). <http://www.redbooks.ibm.com/redbooks/pdfs/sg247973.pdf>.
52. Red Hat Customer Content Services. Getting Started with Red Hat Business Optimizer (2024). https://access.redhat.com/documentation/en-us/red_hat_decision_manager/7.1/html/getting_started_with_red_hat_business_optimizer/index.
53. Agiloft Inc. Agiloft Reference Manual (2022). <https://www.agiloft.com/documentation/agiloft-developer-guide.pdf>.
54. The Data Appeal. Geographic Data (2024). <https://datarade.ai/data-products/the-data-appeal-geographic-data-api-dataset-251m-poi-m-the-data-appeal-company>.
55. Insider Media Ltd. Top Companies Database (2024). <https://www.insidermedia.com/membership/sample-data>.
56. Extensions Hub Yellow Pages Scraper (2024). <https://chromewebstore.google.com/detail/yellow-pages-scraper-yell/mmibhnbhahgckfofpindpgaphgocbkhi?pli=1>.
57. DataForSEO. Amazon Product Database (2024). <https://dataforseo.com/databases/amazon-database>.