PriMera Scientific Engineering
Volume 9 Issue 3 September 2026
DOI: 10.56831/PSEN-09-290
ISSN: 2834-2550

PriMera Scientific Publications

(Elementary) Mathematical Data Models of the Relational and Entity-Relationship Ones

Type: Research Article
Received: July 29, 2026
Published: September 02, 2026

Citation:
Christian Mancas. "(Elementary) Mathematical Data Models of the Relational and Entity-Relationship Ones". PriMera Scientific Engineering 9.3 (2026): 02-17.

Christian Mancas*

Mathematics and Computer Science Department, Ovidius University at Constanta, Romania

***Corresponding Author:** Christian Mancas, Ovidius University, Bd. Mamaia 124, Constanta, CT, Romania.

Abstract

This paper presents (Elementary) Mathematical Data Models of the Relational and the Entity-Relationship Data Models. The first one was obtained starting with a Relational Data Model of itself shown in our *Conceptual Data Modeling and Database Design book (Volume 1: The Shortest Advisable Path)* and then applying several forward and reverse engineering algorithms between these two data models, as well as the Entity-Relationship one. The second one was obtained directly from the Entity-Relationship Data Model of itself also shown in our book.

Keywords: Conceptual Data Modeling; Relational Data Model; Entity-Relationship Data Model; E-R diagram cycle analysis; (Elementary) Mathematical Data Model; Relational and Entity-Relationship Database Management Systems design

Abbreviations

(E)MDM = (Elementary) Mathematical Data Model.
db(s) = database(s).
DFS = depth-first search.
DKNF = domain-key normal form.
E-RD = Entity-Relationship Diagram.
E-RDBMS(es) = Entity-Relationship Database Management System(s).
E-RDM = Entity-Relationship Data Model.
e.g. = for example.
FDM = Functional Data Model.
i.e. = that is.
rdb(s) = relational database(s).
RDBMS(es) = Relational Database Management System(s).
RDM = Relational Data Model.

Introduction

The Relational Data Model (RDM) [1-3] is still the most used one for managing structured data, with powerful Relational Database Management Systems (RDBMSes), like IBM Db2 [4], Oracle Database [5], MS SQL Server [6], etc. based on it.

Unfortunately, even these powerful RDBMSes allow for theoretically nonsense constructs, e.g., declaring superkeys alongside unique keys for any database (db) table. For example, considering a table *COUNTRIES* having unique keys *CountryName* : *COUNTRIES* ↔ ASCII(255) total, *CountryCode* : *COUNTRIES* ↔ ASCII(8) total, and Capital : *COUNTRIES* ↔ *CITIES*, and some other attributes like *PhoneCode* : *COUNTRIES* → NAT(6), *Population* : *COUNTRIES* → NAT(10), etc., you may declare as unique keys in it not only the one-to-one *CountryName*, *CountryCode*, and *Capital*, but also products between them, like *CountryName* · *CountryCode* : *COUNTRIES* ↔ ASCII(255) × ASCII(8), as well as products between them and any other attribute, e.g., *CountryName* · *Population* : *COUNTRIES* ↔ ASCII(255) × NAT(10).

Of course, all above products are one-to-one, as the product of any one-to-one function with any other one is one-to-one, but they are not minimally one-to-one (i.e., are called *superkeys* in the RDM), because you can eliminate one of their members and the remaining function (subproduct) is still one-to-one (e.g., any of the two in *CountryName* · *CountryCode*, as well as *Population* from *CountryName* · *Population*).

Consequently, declaring superkeys in any RDBMS should not be allowed, as they are not only superfluous, but also taking up unnecessary disk space and memory for the index files that are implementing them, as well as slowing down updates to corresponding tables, as the corresponding index files must be updated too.

This is the main reason why we consider that a precise mathematical model of the RDM, like the one we are presenting in this paper, would be beneficial not only to RDBMS designers and developers, but also to their end-users, be they db architects or software developers. There are other reasons as well, out of which the second as importance is the lack of theoretical definitions in the RDM for the tuple constraints.

Moreover, we are convinced that a precise mathematical model of the Entity-Relationship Data Model (E-RDM) [3, 7, 8], which is generally used in the first step of designing relational databases (rdbs), would also be of great help, both for understanding and better using it, as well as for Entity-Relationship Database Management Systems (E-RDBMSes) designers and developers.

The main contributions of this paper are the mathematical models of the RDM and the E-RDM expressed in our (Elementary) Mathematical Data Model ((E)MDM) [9]. These models consider only the initial RDM, as introduced in [1], without its object-oriented later enhancements [4-6, 10], as well as the initial E-RDM, as introduced in [7], without its later enhancements [8].

Related Work

In [3], we presented a RDM model of the E-RDM (see Section 3.3.2), an E-RDM one of itself (see Section 2.9), another E-RDM one of the RDM (see Section 3.7.3), as well as a RDM one of itself (see Section 3.7.1), except for the tuple constraints that were left as Exercises to the readers. To our knowledge, there are no other mathematical data models of either the RDM or the E-RDM.

Materials and Methods

RDM model

First, to obtain the (E)MDM schema of the RDM, we solved Exercise 3.7.5 from [3] to add the *TUPLE_CONSTRAINTS* and *TC_ATTRIBS* domain-key normal form (DKNF) tables to the relational metamodel of the RDM presented in [3] (see Section 3.7.2). Figure 1 shows their schemes and a plausible valid instance for them.

Then, we applied to this extension the *REA*1-2 algorithm from [3] (see Section 3.5) to reverse engineer it into its corresponding E-R diagram (E-RD), shown in Figure 2.

TUPLE_CONSTRAINTS (*x*, *TCExpr* • **Relation*)

x	*TCExpr*	**Relation*
Im(*CONSTRAINTS.x*)	ASCII(255)	= *ConstrRelation* ° *x*
NOT NULL	NOT NULL	
34	*minValue* < *maxValue*	3

TC_ATTRIBS (*x*, *Attribute* • *TupleCnstr*)

x	*TupleCnstr*	*Attribute*
AUTON(8)	*Im*(*TUPLE_CONSTRAINTS.x*)	*Im*(*ATTRIBUTES.x*)
NOT NULL	NOT NULL	NOT NULL
1	34	11
2	34	34

Figure 1: DKNF RDM scheme for storing tuple constraints.

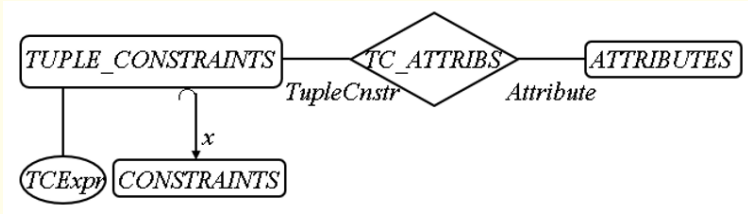

Figure 2: E-RD of tuple constraints.

By applying the depth-first search (DFS) algorithm from [12] to the full final E-RD thus obtained, the twelve cycles shown in Figures 3 to 14 were discovered and classified (out of which 2 of length 2 -one of type circular and one of type commutative-, 8 of length 4 -6 of type commutative and 2 of type general-, and 2 of length 5 - both of type commutative). Then, the *MatBase* E-RD Cycle Analysis Assistance Algorithm from [13] was used to analyze these 12 cycles and detect all corresponding associated constraints, as follows.

The cycle from Figure 3 has length 2 and is of type commutative. Trivially, as no foreign key column should ever reference itself, this diagram anti-commutes; moreover, as no attributes should reference themselves neither directly, nor indirectly, the product *FKAttribute* · *RefAttribute* should be acyclic (which implies that it is asymmetric, hence irreflexive too), which is formalized already in this db scheme by the constraint C_{11} of Fig. 3.21 from [3].

The cycle from Figure 4 has length 2 and is of type circular. Considering any relation *r* and its primary key *k*, constraint *k* is trivially one of *r*'s, i.e., *ConstrRelation* ° *PrimaryKey* is reflexive, (i.e., this diagram locally commutes in *RELATIONS*), i.e., *ConstrRelation*(*PrimaryKey*(*r*)) = *r*, ∀*r*∈*RELATIONS*, which is already formalized in this db scheme by constraint C_4 of Fig. 3.21 from [3]. Dually, considering any constraint *c* of a relation *r* having primary key *k*, *c* may be *k* or not (as *r* may have other constraints as well), so no constraint is associated with node *CONSTRAINTS*.

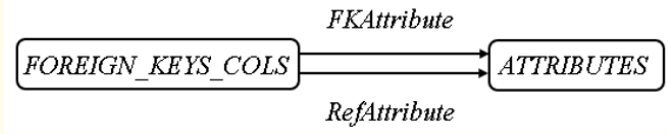

Figure 3: The first cycle from the E-RD in Fig. 3.22 of [3].

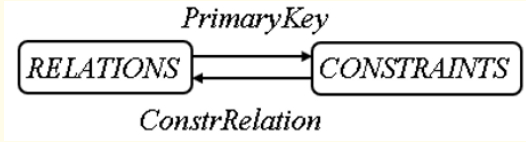

Figure 4: The second cycle from the E-RD in Fig. 3.22 of [3].

The cycle from Figure 5 has length 4, is of type commutative, has source *FOREIGN_KEYS_COLS*, and destination *CONSTRAINTS*.

Considering any foreign key *fk* and any of its columns *fkc* of a relation *r* having primary key *k*, generally, $k \neq fk$, but for tables corresponding to subsets, their primary key may be a foreign one as well, implementing a canonical inclusion, e.g., $x : TUPLE\_CONSTRAINTS \leftrightarrow CONSTRAINTS$ from Figures 1 and 2. Consequently, this diagram neither commutes, nor anti-commutes and, as there is no other object constraint associated with it, corresponds to an uninteresting cycle.

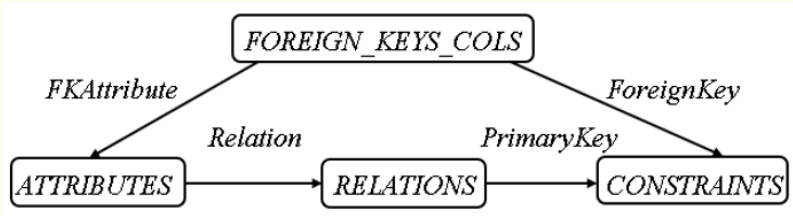

Figure 5: The third cycle from the E-RD in Fig. 3.22 of [3].

The cycle from Figure 6 has length 4, is of type commutative, has source *FOREIGN_KEYS_COLS*, and destination *CONSTRAINTS*.

Considering any foreign key *fk* and any of its referenced columns *fkc* of a relation *r* having primary key *k*, trivially $k \neq fk$ (as not only *k* and *fk* have different types, but, generally, *fk* is a constraint from another table -the referenced one- than the one of *k*), hence this diagram anti-commutes and the following constraint should be added to this db scheme:

$\boldsymbol{C_{18}}$***:*** $(\forall x \in FOREIGN\_KEYS\_COLS)(ForeignKey(x) \neq PrimaryKey(Relation(RefAttribute(x))))$ (No foreign key column may be the primary key of the referenced table.)

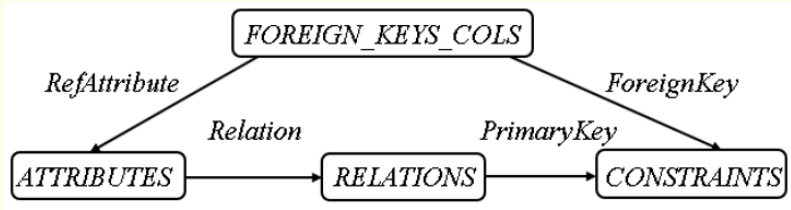

Figure 6: The fourth cycle from the E-RD in Fig. 3.22 of [3].

The cycle from Figure 7 has length 4, is of type commutative, has source *FOREIGN_KEYS_COLS*, and destination *RELATIONS*.

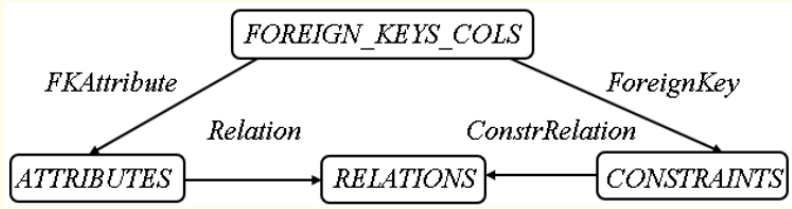

Figure 7: The fifth cycle from the E-RD in Fig. 3.22 of [3].

Considering any foreign key *fk* and any of its columns *fkc* of a relation *r*, trivially *fk* is a constraint of *r*, hence this diagram commutes, which is already formalized in this db scheme by constraint C_8 of Fig. 3.21 from [3] (any attribute of a foreign key should belong to the relation to which the corresponding referential integrity constraint is associated).

The cycle from Figure 8 has length 4, is of type commutative, has source *FOREIGN_KEYS_COLS*, and destination *RELATIONS*.

Considering any foreign key *fk* and any of its referenced columns *fkc* of a relation *r*, trivially *fk* is a constraint of *r* if and only of the corresponding function *fkc* is a self-map, hence this diagram neither commutes, nor anti-commutes.

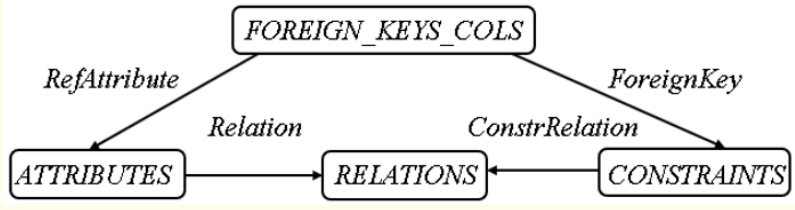

Figure 8: The sixth cycle from the E-RD in Fig. 3.22 of [3].

The cycle from Figure 9 has length 4, is of type general, has sources *KEYS_COLS* and *FOREIGN_KEYS_COLS*, and destinations *ATTRIBUTES* and *CONSTRAINTS*.

Considering any key *k* and any of its columns *kc*, as well as any foreign key *fk* and any of its columns *fkc*, obviously it is possible that *kc* is also part of *fk*, but this is not compulsory.

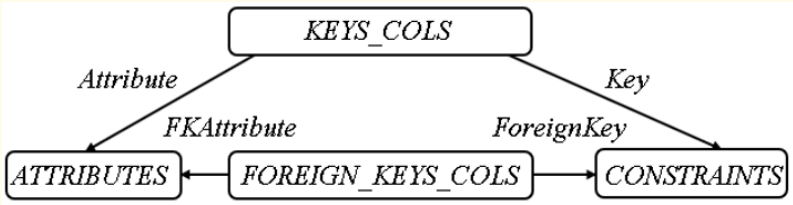

Figure 9: The seventh cycle from the E-RD in Fig. 3.22 of [3].

The cycle from Figure 10 has length 4, is of type general, has sources *KEYS_COLS* and *FOREIGN_KEYS_COLS*, and destinations *ATTRIBUTES* and *CONSTRAINTS*.

Similar to the cycle above, considering any key k and any of its columns kc, as well as any foreign key fk and any of its referenced columns fkc, obviously it is possible that kc is also part of fk, but this is not compulsory, while, always, $k \neq fk$ (as k and fk have different types and, moreover, except when fk is a self-map, they belong to different relations), which is already enforced by constraint C_{18}.

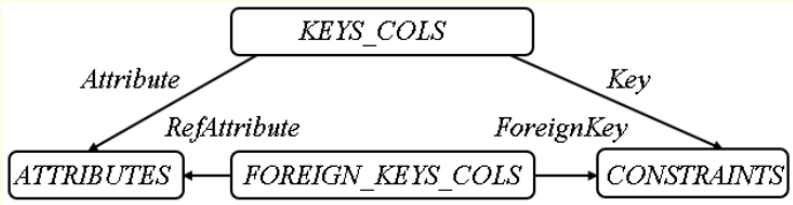

Figure 10: The eighth cycle from the E-RD in Fig. 3.22 of [3].

The cycle from Figure 11 has length 4, is of type commutative, has source *KEYS_COLS*, and destination *CONSTRAINTS*.

Considering any key k and any of its columns kc of a relation r having primary key pk, trivially k may be pk, but this is not compulsory, which means that this cycle is uninteresting.

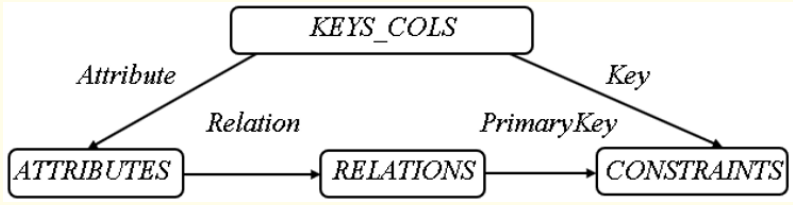

Figure 11: The nineth cycle from the E-RD in Fig. 3.22 of [3].

The cycle from Figure 12 has length 4, is of type commutative, has source *KEYS_COLS*, and destination *RELATIONS*.

Considering any key *k* and any of its columns *kc* of a relation *r*, trivially *k* is a constraint of *r*, which means that this cycle commutes, which is already formalized in this db scheme by the constraint C_5 of Fig. 3.21 from [3] (any attribute of a key should belong to the relation to which the corresponding key constraint is associated).

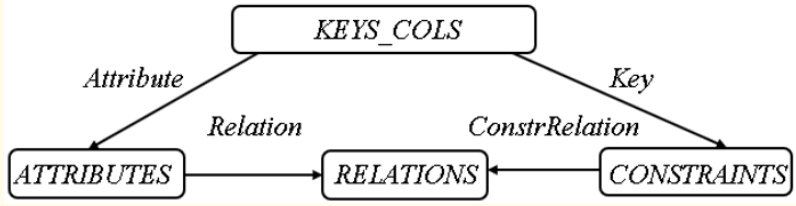

Figure 12: The tenth cycle from the E-RD in Fig. 3.22 of [3].

The cycle from Figure 13 has length 5, is of type commutative, has source *TC_ATTRIBS*, and destination *RELATIONS*.

Considering any tuple constraint *tc* of a relation *r* and any of its columns *tcc*, trivially, by definition, *tcc* must be a column of *r*; hence, this diagram commutes and the following constraint should be added to this db scheme:

$\boldsymbol{C_{19}}$: $(\forall x \in TC\_ATTRIBS)(ConstrRel \circ x \circ TupleCnstr = Relation \circ Attribute)$ (All attributes of a tuple constraint associated to a relation must belong to that relation).

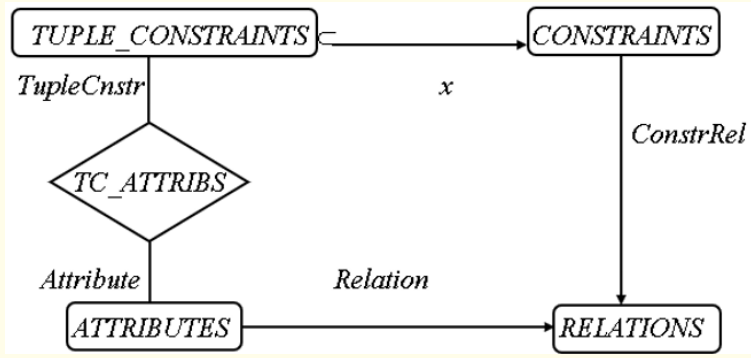

Figure 13: The eleventh cycle from the E-RD in Fig. 3.22 of [3] combined with Figure 2 above.

Finally, the cycle from Figure 14 has length 5, is of type commutative, has source *TC_ATTRIBS* and destination *ATTRIBUTES*.

Considering any tuple constraint *tc* of a relation *r* having primary key *pk* and any of its columns *tcc*, trivially, there may be the case that *pk* takes part in *tc*, but this is not compulsory; hence, this cycle is uninteresting.

In the end, Algorithm *A*1 from [11] was used to translate the final obtained E-R data model of the RDM into (E)MDM, which is shown in Figure 16.

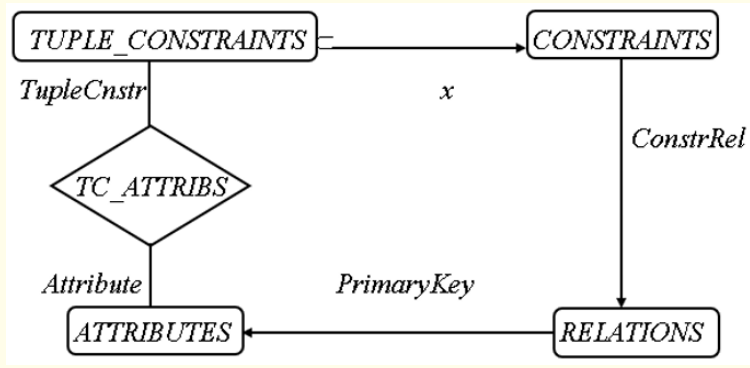

Figure 14: The twelfth cycle from the E-RD in Fig. 3.22 of [3] combined with Figure 2 above.

E-RDM model

To obtain the (E)MDM schema of the E-RDM, we started with the E-RDM of itself presented in [3] (see Section 2.9). First, we simplified the E-RD from Fig. 2.17 of [3], as shown in Figure 15, by adding a set *SETS* and a few user and system attributes to it and *MAPPINGS*, as follows:

- **ValueSet?* : *SETS* → BOOLE, storing *True* whenever the corresponding set is a range one and *False* otherwise (i.e., taking values from *OBJECT_SETS*);
- *CartesianProjection?* : *MAPPINGS* → BOOLE, storing *True* whenever the corresponding mapping is a role of a relationship-type object set and *False* otherwise;
- **Attribute?* : *MAPPINGS* → BOOLE, storing *True* whenever the corresponding mapping is an attribute (i.e., taking values from RANGES) and *False* otherwise.

Consequently, the following subsets become computable:

- **ATTRIBUTES* = {$f \in$ *MAPPINGS* | **Attribute?*(f)}
- **FUNCTIONAL_RELATIONSHIPS* = {$f \in$ *MAPPINGS* | ¬**Attribute?*(f)} = **STRUCTURAL_FUNCTIONS*
- **RANGES* = {$s \in$ *SETS* | **ValueSet?*(s)}
- **OBJECT_SETS* = {s ∈ SETS | ¬*ValueSet?(s)}
- **NON_FUNCTIONAL_RELATIONSHIP_TYPES* = {$s \in$ **OBJECT_SETS* | (∃$f \in$**FUNCTIONAL_RELATIONSHIPS*) (*CartesianProjection?*(f))}
- **ENTITY_TYPES* = **OBJECT_SETS* – **NON_FUNCTIONAL_RELATIONSHIP_TYPES*
- **UNDERLYING_SETS* = {$s \in$ **OBJECT_SETS* | (∃$f \in$**FUNCTIONAL_RELATIONSHIPS*) (*CartesianProjection?*(f) ∧ s = *Codomain*(f))}.

Obviously, the following restrictions must be added to this simplified E-RD model:

- *Description, MaxCard, OneToOne?, Compulsory?, Codomain,* and *CartesianProjection?* are non-prime.
- *Domain* is onto (i.e., any object set should have at least one attribute or structural mapping defined on it).
- *RR5:* In any range, *MinValue* should be less than *MaxValue*.
- *RM8:* Attributes may not be roles (Cartesian projections).
- *RM9:* By definition, any role (Cartesian projection) is totally defined.
- *RM10:* Any relationship-type object set should be a set.
- *RM11:* Mapping domains are object sets (not ranges).

No other keys were obtained by applying Algorithm *A*3 from [14].

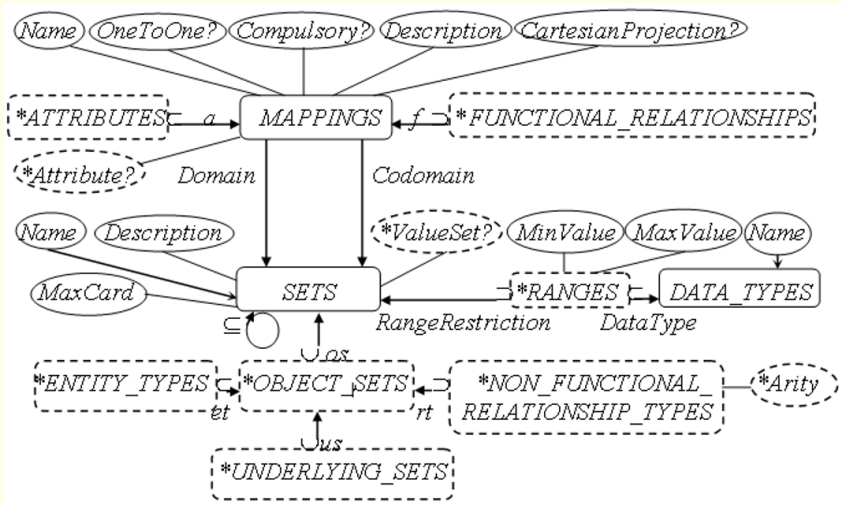

Figure 15: Simplified E-RD of the E-RDM.

Then, by applying the DFS algorithm from [12] to this modified E-RD, only the commutative-type cycle of length 2 made of *Domain* and *Codomain* is obtained. Applying the *MatBase* E-RD Cycle Analysis Assistance Algorithm from [13] did not detect any associated constraint with it: it should neither commute (as, generally, not every structural function is a self-map), nor anti-commute (as there are self-maps), and the non-recursively defined relationship-types constraint is enforced by *RU6*.

In the end, Algorithm A1 from [11] was used to translate the final obtained E-RDM data model of the E-RDM into (E)MDM, shown in Figure 17.

Results and Discussion

RDM model

The (E)MDM scheme of the RDM is shown in Figure 16.

We replaced the *DConstr* attribute of *ATTRIBUTES* with the attributes *minValue*, having id 11 in Table 3.10 from [3] (i.e., the one of *DConstr*) and *maxValue*, having id 34 in that table (storing the minimum and maximum valid values, respectively, of the attributes for which their domain is too large).

Obviously, this model should also contain the tuple constraint C_{15}: *minValue* < *maxValue* (having id 34 in table *CONSTRAINTS* - see Table 3.11 from [3], Fig. 3.9.4).

To also model the materialized views, we added to *RELATIONS* an attribute *RelationExpr* for storing the SQL expressions with which such views are computed.

To model computed columns of fundamental tables too, we also added to *ATTRIBUTES* an attribute *Expression* for storing the corresponding computing expressions.

Moreover, although this attribute is not theoretically needed, as it is provided by all commercial RDBMSes and it also simplifies some constraints, we added to *ATTRIBUTES* a Boolean attribute *Unique?* (storing *True* whenever the corresponding attribute is a key and *False* otherwise).

DATABASES (The set of dbs managed by a RDBMS)

$x \leftrightarrow$ AUTON(4) *total*

DBName $\leftrightarrow$ ASCII(255) *total* (Dbs managed by a RDBMS must have distinct mandatory names.)

DOMAINS (The set of data types provided by a RDBMS)

$x \leftrightarrow$ AUTON(2) *total*

DomainName $\leftrightarrow$ ASCII(32) *total* (Domains must have unique mandatory names.)

RELATIONS (The set of tables and materialized views managed by the RDBMS)

$x \leftrightarrow$ AUTON(8) *total*

RelationName $\rightarrow$ ASCII(255) *total*

RelationExpr $\rightarrow$ ASCII(4096) (SQL expression for computing a materialized view)

Database : *RELATIONS* $\rightarrow$ *DATABASES total*

*RR*3: *RelationName* • *Database key* (Relations of a db must have distinct names.)

*RR*4: *RelationExpr* • *Database key* (There is no use of storing in a db two materialized views having same computing expression.)

Figure 16: The (E)MDM model of the RDM.

ATTRIBUTES (The set of columns of the relations managed by the RDBMS)

$x \leftrightarrow$ AUTON(10) *total*

AttributeName $\rightarrow$ ASCII(255) *total*

minValue $\rightarrow$ ASCII(255) *nonprime*

maxValue $\rightarrow$ ASCII(255) *nonprime*

Total? $\rightarrow$ BOOLE *total*, *nonprime*

Unique? $\rightarrow$ BOOLE *total*, *nonprime*

DefaultValue$\rightarrow$ ASCII(255) *nonprime*

Expression $\rightarrow$ ASCII(255)

Relation : *ATTRIBUTES* $\rightarrow$ *RELATIONS total*, *onto* (Any relation must have at least one attribute.)

Domain : *ATTRIBUTES* $\rightarrow$ *DOMAINS total*

*RA*4: *AttributeName* • *Relation key* (Attributes of a relation must have distinct names.)

*RA*5: $(\forall x \in ATTRIBUTES)(DefaultValue(x) \in \text{NULLS} \vee (DefaultValue(x) \in Domain(x) \wedge (minValue(x) \in \text{NULLS} \vee minValue(x) \leq DefaultValue(x)) \wedge (maxValue(x) \in \text{NULLS} \vee DefaultValue(x) \leq maxValue(x))))$(Default values must belong to the attributes' domains.)

*RA*6: $(\forall x \in ATTRIBUTES)(minValue(x) \in \text{NULLS} \vee minValue(x) \in Domain(x))$(Minimum values must belong to the attributes' domains.)

Figure 16: (Continued).

*RA*7: $(\forall x \in ATTRIBUTES)(maxValue(x) \in \text{NULLS} \vee maxValue(x) \in Domain(x))$(Maximum values must belong to the attributes' domains.)

*RA*8: *Expression* • *Relation key* (There should not be two computed attributes of a relation having the same computing expression.)

C_{15}: $(\forall x \in ATTRIBUTES)(minValue(x) < maxValue(x))$(The minimum value of any domain constraint must be less than its maximum one.)

CONSTRAINTS (The set of relational constraints of the relations managed by the RDBMS)

$x \leftrightarrow$ AUTON(10) *total*

ConstrName $\rightarrow$ ASCII(255) *total*

ConstrType $\rightarrow$ {'F', 'K', 'T'} *total* (Any explicit constraint has type either foreign key ('F'), or (unique) key ('K'), or tuple ('T').)

ConstrRelation : *CONSTRAINTS* $\rightarrow$ *RELATIONS total*

**DB* = *Database* ° *ConstrRelation*

*RC*4: *ConstrName* • **DB key* (Constraints of a db must have distinct names.)

Figure 16: (Continued).

FOREIGN_KEYS_COLS (The set of triples <*fk*, *c*, *rc*> storing the fact that column *c* referencing column *rc* takes part in foreign key *fk*)

$x \leftrightarrow$ AUTON(9) *total*

Position $\rightarrow$ [1, 64] *total*

ForeignKey : *FOREIGN_KEYS_COLS* $\rightarrow$ *CONSTRAINTS total*

FKAttribute : *FOREIGN_KEYS_COLS* $\rightarrow$ *ATTRIBUTES total*

RefAttribute : *FOREIGN_KEYS_COLS* $\rightarrow$ *ATTRIBUTES total*

*RFK*3: *FKAttribute* • *ForeignKey key* (No attribute should appear more than once in a foreign key.)

*RFK*4: *ForeignKey* • *Position key* (There may not be more than one attribute in any position of a concatenated foreign key.)

*RFK*5: *RefAttribute* • *ForeignKey key* (No attribute should be referenced more than once in a foreign key.)

C_{18}: $(\forall x \in FOREIGN\_KEYS\_COLS)(ForeignKey(x) \neq PrimaryKey(Relation(RefAttribute(x))))$

(No foreign key column may be the primary key of the referenced table.)

Figure 16: (Continued).

KEYS_COLUMNS (The set of pairs <*k*, *c*> storing the fact that key *k* (also) contains column *c*.)

$x \leftrightarrow$ AUTON(9) *total*

Position $\rightarrow$ [1, 64] *total*

Key : *KEY_COLUMNS* $\rightarrow$ *CONSTRAINTS total*

Attribute : *KEY_COLUMNS* $\rightarrow$ *ATTRIBUTES total*

*RK*3: *Attribute* • *Key key* (No attribute should appear more than once in a (unique) key.)

*RK*4: *Key* • *Position key* (There may not be more than one attribute in any position of a concatenated key.)

C_1: $(\forall x \in RELATIONS)(PrimaryKey(x) \notin \text{NULLS} \Rightarrow ConstrType(PrimaryKey(x)) = \text{'K'})$ (Primary keys are constraints of type *key*.)

C_2: $(\forall x \in RELATIONS)(\forall y \in KEYS\_COLUMNS)\ (PrimaryKey(x) = Key(y) \Rightarrow Relation \circ Attribute(y) = x)$ (Any attribute of the primary key of any relation must be an attribute of that relation.)

C_3: $(\forall x \in CONSTRAINTS)(Key^{-1}(\{x\}) = \{c_1, \ldots, c_n\}, n > 1, \text{natural} \Rightarrow Attribute(c_1) \bullet \ldots \bullet Attribute(c_n)$ one-to-one $\wedge\ Attribute(c_1) \bullet \ldots \bullet Attribute(c_{i-1}) \bullet Attribute(c_{i+1}) \bullet \ldots \bullet Attribute(c_n)$ not one-to-one, $1 \leq i \leq n)$(Concatenated keys should be minimally one-to-one, i.e., they should be keys, not superkeys.)

C_4: *ConstrRelation* $\circ$ *PrimaryKey reflexive* (The primary key of any relation is a constraint of that relation.)

Figure 16: (Continued).

C_5: *ConstrRelation* $\circ$ *Key* = *Relation* $\circ$ *Attribute* (Any attribute of a key should belong to the relation to which the corresponding key constraint is associated.)

C_6: $(\forall x \in KEYS\_COLUMNS)(PrimaryKey(Relation(Attribute(x))) = Key(x) \Rightarrow Attribute(x) \in PrimaryKey(Relation(Attribute(x)))$ (Primary key attributes are key attributes.)

C_7: $(\forall x \in FOREIGN\_KEYS\_COLS)(ConstrType(ForeignKey(x)) = \text{'F'})$ (Referential integrities are constraints of type *foreign key*.)

C_8: *ConstrRelation* $\circ$ *ForeignKey* = *Relation* $\circ$ *FKAttribute* (Any attribute of a foreign key should belong to the relation to which the corresponding referential integrity constraint is associated.)

C_9: $(\forall x,y \in FOREIGN\_KEYS\_COLS, x \neq y)\ (ForeignKey(x) = ForeignKey(y) \Rightarrow Relation \circ RefAttribute(x) = Relation \circ RefAttribute(y))$ (All attributes referenced by a foreign key should belong to a same relation.)

C_{10}: $(\forall x \in FOREIGN\_KEYS\_COLS)(Im(FKAttribute(x)) \subseteq Im(RefAttribute(x)))$ (Referential integrity enforcement: values of foreign keys should always be among those of the corresponding referenced columns.)

C_{11}: $(\forall x \in FOREIGN\_KEYS\_COLS)\ ((FKAttribute \bullet RefAttribute)(x)\ acyclic)$ (No attribute should reference itself, neither directly, nor indirectly.)

C_{12}: $(\forall x \in CONSTRAINTS)(\exists y \in FOREIGN\_KEYS\_COLS)(ConstrType(x) = \text{'F'} \Rightarrow ForeignKey(y) = x)$ (Any foreign key has at least one attribute)

C_{13}: $(\forall x \in CONSTRAINTS)(\exists y \in KEYS\_COLUMNS)(ConstrType(x) = \text{'K'} \Rightarrow Key(y) = x)$ (Any (unique) key has at least one attribute.)

Figure 16: (Continued).

C_{16}: $(\forall x,y \in KEYS\_COLUMNS, x \neq y)(Key(x) = Key(y) \Rightarrow \neg Unique?(x) \wedge \neg Unique?(y))$(Single keys should not take part in concatenated ones.)

C_{17}: $(\forall x \in CONSTRAINTS)(\forall y \in RELATIONS)(PrimaryKey(y) = x \wedge Key^{-1}(\{x\}) = \{c_1, \ldots, c_n\}, n > 0,$ natural $\Rightarrow \neg Null?(Attribute(c_1)) \wedge \ldots \wedge \neg Null?(Attribute(c_n)))$ (Primary key columns are totally defined, i.e., not null.)

TUPLE_CONSTRAINTS $\subseteq$ ***CONSTRAINTS*** (The subset of tuple (check) constraints)

x : *TUPLE_CONSTRAINTS* $\leftrightarrow$ *CONSTRAINTS total*

TCExpr $\rightarrow$ ASCII(255) *total*

**Relation = ConstrRelation* $\circ$ x

TCK: *TCExpr* $\bullet$ **Relation key* (It would make no sense to have two tuple constraints of a same relation having same expression.)

C_{14}: $(\forall x \in TUPLE\_CONSTRAINTS)(ConstrType(x) =$ 'T')(Tuple constraints' type is 'T'.)

TC_ATTRIBS = (TupleCnstr $\rightarrow$ ***TUPLE_CONSTRAINTS, Attribute*** $\rightarrow$ ***ATTRIBUTES)*** (The set of pairs of type $<tc, a>$ storing the fact that attribute a is mentioned in tuple constraint tc)

$x \leftrightarrow$ AUTON(8) *total*

C_{19}: $(\forall x \in TC\_ATTRIBS)(ConstrRel \circ x \circ TupleCnstr = Relation \circ Attribute)$ (All attributes of a tuple constraint associated to a relation must belong to that relation.)

Figure 16: (Continued).

Finally, again as this type of constraint is provided as well by all commercial RDBMSes (although it is not part of the RDM theory, but is very useful for end-users), we also added to *ATTRIBUTES* a text attribute *DefaultValue* storing the expression for computing the corresponding default values.

As in this scheme there are neither incoherent, nor redundant constraint sets, by applying Algorithms *ACME*, *AACE*, and *AME* from [15] nothing is modified.

However, as those algorithms do not consider object constraints, in fact, C_3 implies C_{16}, but we are accepting this redundancy academically, for making things clearer.

Please note that, unfortunately, as we have already mentioned in [3], commercial RDBMSes are not enforcing constraints *RR4, RA8, RFK3, RFK5, RK3, C3, C11, C16*, and *TCK*.

E-RDM model

The (E)MDM scheme of the E-RDM is shown in Figure 17.

As in this scheme there are neither incoherent, nor redundant constraint sets, by applying Algorithms *ACME, AACE*, and *AME* from [15] nothing is modified either.

Please note that, as expected [16], the E-RDM may be basically defined using only three sets (*SETS, DATA_TYPES*, and *MAPPINGS*), i.e., its expressive power is exactly the one of the Functional Data Model (FDM) [17, 18], i.e., the only fundamental relationship-type sets are the functional ones.

(***SETS***, ⊆) (The partially ordered by inclusion collection of sets of interest)

$x \leftrightarrow$ AUTON(38) *total*

Name $\leftrightarrow$ ASCII(255) *total*

Description $\rightarrow$ ASCII(255) *total non-prime*

MaxCard $\rightarrow$ NAT(38) *total non-prime*

**ValueSet?* = *True* for ranges and *False* for object sets

****OBJECT_SETS*** = $\{s \in SETS \mid \neg$ **ValueSet?*$(s)\}$

(***DATA_TYPES***, ⊆) (The partially ordered by inclusion set of data types of interest)

$x \leftrightarrow$ AUTON(2) *total*

Name $\leftrightarrow$ ASCII(255) *total*

****RANGES*** = $\{s \in SETS \mid$ **ValueSet?*$(s)\}$

$x \leftrightarrow$ AUTON(4) *total*

MinValue $\rightarrow$ ASCII(255) *total*

MaxValue $\rightarrow$ ASCII(255) *total*

DataType : *RANGES* $\rightarrow$ *DATA_TYPES total*

RS: *SETS* = **OBJECT_SETS* $\oplus$ **RANGES*

Figure 17: The (E)MDM scheme of the E-RDM.

*RR*4: *DataType* • *MinValue* • *MaxValue key* (There may not be two ranges of a same data type having the same minimum and maximum values.)

*RR*5: $(\forall r \in RANGES)(MinValue(r) < MaxValue(r))$ (In any range, *MinValue* should be less than *MaxValue*.)

MAPPINGS (The set of mapping of interest – be they attributes or functional relationships)

$x \leftrightarrow$ AUTON(38) *total*

Name $\rightarrow$ ASCII(255) *total*

Description $\rightarrow$ ASCII(255) *non-prime*

OneToOne? $\rightarrow$ BOOLE *total non-prime*

Compulsory? $\rightarrow$ BOOLE *total non-prime*

CartesianProjection? $\rightarrow$ BOOLE *total non-prime*

**Attribute?* = *True* for *Codomain*$(x) \in$ *RANGES* and *False* otherwise

Domain : *MAPPINGS* $\rightarrow$ *SETS total onto*

Codomain : *MAPPINGS* $\rightarrow$ *SETS total non-prime*

*RM*6: *Name* • *Domain key* (There may not be two mappings having same names and domains.)

*RU*5: $(\forall f \in MAPPINGS)(\exists g \in MAPPINGS)$ $(CartesianProjection(f) \Rightarrow f \neq g \wedge CartesianProjection(g) \wedge Domain(f) = Domain(g))$ (Any non-functional relationship set must have at least two roles.)

Figure 17: (Continued).

*RU*6: $(\forall f_i \in MAPPINGS)(\forall 1 \leq i \leq n > 0$, naturals$)(CartesianProjection(f_i) \wedge Codom(f_i) = Dom(f_{i+1}) \Rightarrow Dom(f_1) \neq Codom(f_n))$ (Hierarchies of non-functional relationship types should be acyclic. Note that, for $n = 1$, this means that roles may not be self-maps, i.e., recursively defined non-functional relationship-type sets are not allowed.)

*RM*8: $*Attribute? \neg|\!\!-\!\!- CartesianProjection?$(Attributes may not be roles).

*RM*9: $CartesianProjection? \neg|\!\!-\!\!- Compulsory?$ (By definition, any role is totally defined.)

*RM*10: $(\forall f_i \in MAPPINGS)(\forall 1 \leq i \leq n > 0$, naturals$) (\forall y \in NON_FUNCTIONAL_RELATIONSHIP_TYPES) (CartesianProjection(f_i) \wedge Dom(f_i) = y \Rightarrow f_1 \bullet \ldots \bullet f_n$ *one-to-one*(Any non-functional relationship-type object set should be a set, i.e., the product of its roles must be one-to-one.)

*RM*11: $(\forall f \in MAPPINGS)(d = Domain(f) \Rightarrow \neg *ValueSet(d))$(Mapping domains are object sets.)

$*ATTRIBUTES = \{f \in MAPPINGS \mid *Attribute?(f)\}$

$*FUNCTIONAL\_RELATIONSHIPS = \{f \in MAPPINGS \mid \neg *Attribute?(f)\}$

*RM*12: $MAPPINGS = *FUNCTIONAL\_RELATIONSHIPS \oplus *ATTRIBUTES$

$*NON\_FUNCTIONAL\_RELATIONSHIP\_TYPES = \{s \in OBJECT\_SETS \mid (\exists f \in *FUNCTIONAL\_RELATIONSHIPS)(CartesianProjection?(f))\}$

$*Arity(rt) = card(\{y \in MAPPINGS \mid CartesianProjection(y) \wedge Domain(y) = rt\})$

$*ENTITY\_TYPES = *OBJECT\_SETS \text{ — } *NON\_FUNCTIONAL\_RELATIONSHIP\_TYPES$

*RO*6: $*OBJECT\_SETS = *ENTITY\_TYPES \oplus *NON\_FUNCTIONAL\_RELATIONSHIP\_TYPES$

Figure 17: (Continued).

Conclusion

The main contributions of this paper are the (E)MDM schemas of the RDM and the E-RDM.

We started both from the RDM scheme of itself and the E-RDM data model of itself, respectively, as published in [3], which we first enhanced, then detected and analyzed the cycles in their E-RDs, and finally translated the E-RDM data models into corresponding (E) MDM schemes.

The (E)MDM schema obtained proves once more that the only fundamental relationships needed for conceptual data modeling are the functions (mappings).

We consider that these mathematical data models have not only significant theoretical value, but also a practical one, for both users, designers, and developers of RDBMSes and E-RDBMSes, respectively.

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.

Acknowledgements

This research was not sponsored by anybody and nobody other than its author contributed to it. The author is grateful to Mihaela Virginia Mancas and Diana Christina Mancas, who are always carefully checking all our manuscripts.

References

1. Codd EF. “A relational model for large shared data banks”. CACM 13.6 (1970): 377-387.

2. Abiteboul S, Hull R and Vianu V. “Foundations of Databases”. Addison-Wesley, Reading, MA (1995).
3. Mancas C. “Conceptual Data Modeling and Database Design: A Completely Algorithmic Approach. Volume 1: The Shortest Advisable Path”. Apple Academic Press, Waretown, NJ (2015).
4. IBM Corp. “IBM Db2” (2026).
5. Oracle Corp. “Oracle® AI Database” (2026).
6. Microsoft Corp. “Microsoft SQL Server 2025” (2026).
7. Chen PP. “The entity-relationship model. Toward a unified view of data”. ACM TODS 1.1 (1976): 9-36.
8. Thalheim B. “Entity-Relationship Modeling: Foundations of Database Technology”. Springer-Verlag, Berlin, Germany (2000).
9. Mancas C. “The (Elementary) Mathematical Data Model revisited”. PriMera Scientific Engineering 5.4 (2024): 78-91.
10. Date CJ and Darwen H. “Foundation for future database systems: The third manifesto; a detailed study of the impact of type theory on the relational model of data, including a comprehensive model of type inheritance (2 ed.)”. Addison-Wesley, Reading, MA (2000).
11. Mancas C and Mancas DC. “MatBase Algorithm for Translating Entity-Relationship Data Models into (Elementary) Mathematical Data Model Schemes”. PriMera Scientific Engineering 7.6 (2025): 04-11.
12. Mancas C and Mocanu A. “Matbase DFS Detecting and Classifying E-RD Cycles Algorithm”. J. Comp. Sci. Appl. Inform. Technol 2.4 (2017): 1-14.
13. Mancas C. “MatBase E-RD Cycles Associated Non-Relational Constraints Discovery Assistance Algorithm”. Intelligent Computing. CompCom 2019. Advances in Intelligent Systems and Computing 997 (2019).
14. Mancas C. “Algorithms for Key Discovery Assistance”. BIR 2016, Lecture Notes in Business Information Processing 261 (2016): 322-338.
15. Mancas C. “MatBase Constraint Sets Coherence and Minimality Enforcement Algorithms”. Proc. 22nd ADBIS Conf. on Advances in DB and Inf. Syst., LNCS, Springer, Cham, Switzerland 11019 (2018): 263-277.
16. Mancas C. and Dragomir S. “On the Equivalence Between Entity-Relationship and Functional Data Modeling”. In: Hamza M. (ed.). Proc. 7th IASTED Int. Conf. on Soft. Eng. and App. (SEA 2003) (2003): 335-340.
17. Kerschberg L. and Pacheco JES. “A Functional Data Base Model”. Monographs in Computer Science and Computer Applications No. 2/76 (1976).
18. Shipman DW. “The Functional Data Model and the Data Language DAPLEX”. ACM ToDS 6.1 (1981): 140-173.