Relaxed Functional Dependencies—A Survey of Approaches

Loredana Caruccio, Vincenzo Deufemia, and Giuseppe Polese, *Member, IEEE*

Abstract—Recently, there has been a renovated interest in functional dependencies due to the possibility of employing them in several advanced database operations, such as data cleaning, query relaxation, record matching, and so forth. In particular, the constraints defined for canonical functional dependencies have been relaxed to capture inconsistencies in real data, patterns of semantically related data, or semantic relationships in complex data types. In this paper, we have surveyed 35 of such functional dependencies, providing a classification criteria, motivating examples, and a systematic analysis of them.

Index Terms—Functional dependencies, data quality, axiomatization, approximate match

1 Introduction

Data dependencies have been used to define data integrity constraints, aiming to improve the quality of database schemas and reduce manipulation anomalies [1]. There are several types of data dependencies, including functional [2], multivalued [3], and join dependencies [4]. Among these, functional dependencies (FDs) are the most commonly known, mainly due to their use in database normalization processes.

As the relational data model evolved in different directions, also FD theory underwent several extensions to enable the specification of integrity constraints in new application domains. For instance, functional dependencies were defined for fuzzy data [5], XML structured data [6], [7], multimedia data [8], temporal data [9], and so forth. In particular, fuzzy functional dependencies (FFDs) were based on the fuzzy resemblance relation in place of the 'equality' relation used in the canonical FD [5]. Analogously, similarity functions were used to derive FDs for multimedia data [8], due to the intrinsic difficulty in performing exact comparisons on them. Other than FDs relaxing on the comparison method, we also find examples of FD definitions relaxing on the satisfiability constraint, that is, FDs possibly not satisfied for a subset of data. For this reason, in this paper we will refer to these imprecise functional dependencies as *relaxed functional dependencies* (RFDs), in that they relax one or more constraints of the canonical FD.

RFDs are mainly used in alphanumeric databases, but no longer for schema design purposes, rather to support several new challenging applications, such as query relaxation [10], data cleaning [11], and record matching [12]. In the last decades, several RFDs have been defined, some of which aiming to solve specific problems. However, many RFD definitions lack highlighting similarities and/or advances with respect to existing ones. This motivates the need for a review of existing RFDs, since in the literature we can only find analysis on some specific aspects, such as the types of algorithms for discovering them from data [13]. Thus, we think that a broader comparison is needed, in order to analyze and classify RFDs based on their features and their relationships.

In this paper we survey 35 RFDs, providing some criteria to classify them, and a general semantics to systematically describe and relate them. The goal of the survey is threefold: i) to analyze existing RFDs, highlighting both theoretical and practical issues; ii) to provide a comprehensive and classified list of them, useful for researchers, database designers, and database tool vendors; and, iii) to help users select the RFDs more suitable for their purposes. Although in the literature there are many other types of data dependencies, such as inclusion [14], and multivalued dependencies [3], the paper focuses on relaxations of FDs, and not of other types of data dependencies.

The remainder of this paper is organized as follows. Section 2 provides a detailed description of the application domains motivating the study of RFDs. In Section 3 we introduce the criteria we have used to classify the surveyed RFDs. Such a classification is provided throughout Sections 4, 5, and 6. Section 7 presents a global comparison of the surveyed RFDs, focusing on both theoretical and practical aspects. Finally, conclusions and future work are discussed in Section 8. In order to facilitate the reading of the paper, in Table 1 we list the notations used in the definition of the surveyed RFDs.

2 Motivations

Let us first introduce an example that we will use in the rest of the paper to define RFDs and explain underlying concepts. It concerns a medical database, whose relational schema is shown in Fig. 1.

Before discussing motivations for RFDs, let us first recall the definition of the canonical FD. Consider a relational schema $\mathcal{R}$ defined over a set of attributes $attr(\mathcal{R})$, derived as the union of attributes from relation schemas of $\mathcal{R}$, assuming w.l.o.g.

• *The authors are with the Department of Computer Science, University of Salerno, Via Giovanni Paolo II, nr. 13284084, Fisciano, Italy. E-mail: {lcaruccio, deufemia, gpolese}@unisa.it.*

Manuscript received 1 Apr. 2014; revised 11 May 2015; accepted 4 Aug. 2015. Date of publication 24 Aug. 2015; date of current version 3 Dec. 2015. Recommended for acceptance by Z. Ives.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2015.2472010

TABLE 1
A Summary of Notations

Symbol	Description
$\mathcal{R}$	relational database schema
R	relation schema
r, s	database instance
X, Y	set of attributes
A, B	attribute
$dom(A)$	attribute domain
a, b	attribute value
π_X	partition on a set X
σ_c	tuple selection based on a condition c
$\phi, \theta, \psi, \mu, \varrho, \chi$	function
ϵ, α, β	threshold
Pr	probability
c	constraint or condition
T_r	pattern tableau

they all have unique names. For each attribute $A \in attr(\mathcal{R})$, its domain is specified in $\mathcal{R}$, denoted by $dom(A)$. For an instance r of $\mathcal{R}$ and a tuple $t \in r$, we use $t[A]$ to denote the projection of t onto A; similarly, for a set X of attributes in $attr(\mathcal{R})$, $t[X]$ denotes the projection of t onto X. An FD over a relation schema R of $\mathcal{R}$ is a statement $X \rightarrow Y$ (X implies Y) with $X, Y \subseteq attr(R)$, such that, given an instance r over R, $X \rightarrow Y$ is satisfied in r if and only if for every pair of tuples (t_1, t_2) in r, whenever $t_1[X] = t_2[X]$, then $t_1[Y] = t_2[Y]$.

Nevertheless, there are situations where the canonical definition of FD is not suitable, because we might need to relax some of its properties. For instance, for the medical database in Fig. 1, we can intuitively say that Symptom $\rightarrow$ Diagnosis holds, but we can only expect that tuples for two different patients show "similar" diagnosis when they show "similar" symptoms. Thus, several new dependencies have been introduced to cope with approximate comparisons, such as the FFD [5], [15]. The latter specifies that for any two sets of attributes $X, Y \subseteq attr(R)$, X implies Y, denoted with $X \rightsquigarrow Y$, in a fuzzy relation r on R, if for all pairs of tuples t_1 and t_2 of r, that is, those with membership value $\mu_r(t_i) > 0$, we have $\mu_{eq}(t_1[X], t_2[X]) \leq \mu_{eq}(t_1[Y], t_2[Y])$, where eq is a fuzzy resemblance relation. Thus, it is possible to define a proper eq relation to model an FFD Symptom $\rightsquigarrow$ Diagnosis.

However, there are many contexts in which it is necessary to capture FDs that cannot be simply defined in terms of a greater similarity on the right-hand side (RHS) with respect to the left-hand side (LHS). This is particularly true in the multimedia database context, where it might be necessary to capture dependencies on data by using different similarity functions on the two sides of the FD [8]. As an example, in the medical database of Fig. 1, ECG $\rightarrow$ Pulse is a dependency in which the LHS and the RHS are different media types, hence their similarity needs to be evaluated based on different similarity functions. This is possible through the following *type-M dependency* TMFD: $\text{ECG}_{(\text{FRACTAL},\epsilon')} \rightarrow \text{Pulse}_{(\text{HS},\epsilon'')}$, which models the dependency between the electrocardiography and the heartbeat multimedia attributes by using the image similarity function FRACTAL, with threshold ϵ', to compare electrocardiograms, and the sound similarity function HS, with threshold ϵ'', to compare heart pulses [8].

On the other hand, the constraints of the canonical FD have recently been relaxed also in the context of alphanumeric databases. As an example, new RFDs have been defined for approximate query processing [10] and data cleaning activities [11]. In query processing, problems might arise in case the answer set is empty or too narrow. Thus, the user might be interested in retrieving not only the items exactly matching the original query Q, but also the similar ones, ranked according to their relevance to Q. This is done by rewriting Q in order to stretch the result set. One way to do this is to rewrite the conditions of Q yielding minimal changes in the characteristics of the matching tuples [10], i.e., the conditions defined on attributes with little or no influence on the values of other attributes in a tuple. In [10] these attributes have been detected through *approximate functional dependencies* (AFDs) [16], which are FDs holding on almost all tuples. Thus, the query rewriting is performed on the attributes rarely appearing on the LHS of AFDs [10]. As an example, let us consider the AFDs in Table 2 holding on the Medicine relation of the database in Fig. 1, and the approximate query $Q =$ "Name like Aulin & Cost like5\$". After retrieving the exactly matching tuple, the approach

Clinical Record

Num	Date	Patient	Age	#Room	CheckinWard	CheckoutWard	Hospitalization	ECG	Pulse	Diastolic	Systolic	Symptom	Diet	Surgery	Diagnosis	...
1	2014-03-01	087-34-7789	61	189	Neurology	Neurology	1	Image1.jpg	pulse1.wav	83	130	Tachycardia, breathe shortness	2	-	Panic attack	
2	2014-03-01	087-11-3455	84	300	Cardiology	Neurology	10	Image2.jpg	pulse2.wav	90	140	Dizziness	1	-	Ictus	
3	2014-03-01	089-65-3325	44	267	Cardiology	Cardiology	18	Image3.jpg	pulse3.wav	80	120	Arm ache, nausea	3	Angioplasty	Heart attack	
4	2014-03-02	091-87-9437	9	205	Pediatrician	Pediatrician	4	Image4.jpg	pulse4.wav	77	117	Stomach ache	5	Gastric lavage	Indigestion	
5	2014-03-02	092-12-1439	85	151	Neurology	Neurology	9	Image5.jpg	pulse5.wav	87	134	Dizziness, speech impairment	4	Clipping	Ictus	
6	2014-03-20	088-52-1314	47	262	Cardiology	Cardiology	19	Image6.jpg	pulse6.wav	92	133	Arm ache, breast ache	3	Angioplasty	Heart failure	
...																

Patient

SSN	Name	Birthdate	Gender	Address	BloodType
087-34-7789	Andrea White	1953-03-14	F	987 Jefferson AV, NV	0+
087-11-3455	Mary Brown	1930-08-31	F	555 Fifth AV, NV	A+
089-65-3325	Bill Mc Gregor	1970-12-21	M	100 Canal ST, NJ	0+
091-87-9437	John Smith	2005-11-01	M	321 Delaware AV, CA	AB+
092-12-1439	Eric Ford	1929-10-19	M	774 Vermont ST, WA	A-
...					

Doctor

IdDoctor	Name	Specialization	Role	Experience	Salary	Level	Tax
1	George Johnson	Neurology	Junior Dr.	1 years	\$118,000	E	\$27,140
2	Joe House	Cardiology	Head Physician	10 years	\$314,000	B	\$94,200
3	Derek Williams	Pediatrician	Specialized Dr.	2 years	\$156,000	D	\$39,000
4	Henry Jones	Neurology	Specialized Dr.	3 years	\$158,000	D	\$39,500
5	Victor Sanchez	Radiology	Senior Surgeon	5 years	\$225,000	C	\$63,000
...							

Medicine

Producer	Name	Category	Symptoms	Weight	Cost	ActiveIngredient	Rx
Angelini	Aulin	NSAID	Inflammation	100mg	\$6,5	Nimesulide	Yes
Angelini	Aulin	NSAID	Inflammation	50mg	\$5	Nimesulide	Yes
Dompé	Oki	NSAID	Inflammation	80mg	\$4	Ketoprofen	Yes
Zambon It	Spidifen	NSAID	Headache	200mg	\$5	Ibuprofen	No
Bayer	Aspirin	NSAID	Fever status	400mg	\$4	Acetylsalicyclic acid	No
Bristol lab.	Panadol	NSAID	Osteoarthritis	500mg	\$5	Paracetamol	No
...							

Check

IdCheck	Name	Patient	PrescriptionDate	ExecutionDate	Doctor
1	Cholesterol	087-11-3455	2014-03-03	2014-03-06	1
2	HIV	089-65-3325	2014-03-07	2014-03-14	4
3	HCV	091-87-9437	2014-03-04	2014-03-05	3
4	AIDS	092-12-1439	2014-03-05	2014-03-14	4
5	Mammography	091-87-9437	2014-03-06	2014-03-08	5
6	Echocardiogram	088-52-1314	2014-03-21	2014-03-25	2
...					

Fig. 1. A portion of a medical database instance.

TABLE 2
A Sample of AFDS

AFD	**Confidence**
Name → Producer	97%
Category → Symptoms	86%
Name, Weight → Cost	82%
Name, Cost → Weight	54%

proposed in [10] looks for approximately matching tuples by first rewriting the query condition involving the attribute Cost, since it has a reduced impact on other attribute values with respect to Name, as inferred from the detected AFDS.

In the context of data cleaning, RFDS have been used to detect records referring to the same real world entities [12]. As an example, when building a database of historical clinical activities on patients, by integrating the databases of different hospitals and healthcare services, there might be the need to detect records referring to same patients. However, if some of the source databases to be integrated do not have an unique identifier (e.g., SSN) for distinguishing records of different patients, then we need to evaluate alternative attributes to join related records. To this end, *matching dependencies* (MDS) can be used [12]. As an example, given the relations $\mathsf{R_1(FN, LN, addr, BD, BT, sex)}$ and $\mathsf{R_2(FN, LN, post, BD, BG, sex)}$, the following MD:
$(\mathsf{R_1[LN] = R_2[LN]}) \wedge (\mathsf{R_1[addr] = R_2[post]}) \wedge (\mathsf{R_1[FN]} \approx_{\mathsf{d}} \mathsf{R_2[FN]}) \rightarrow \mathsf{R_1[BD, BT, sex]} \rightleftharpoons \mathsf{R_2[BD, BG, sex]}$
states that if tuples $t_1 \in \mathsf{R_1}$ and $t_2 \in \mathsf{R_2}$ have the same values in the common attribute last name (LN), and in the attributes address (addr) of $\mathsf{R_1}$ and post of $\mathsf{R_2}$, and have similar values in the common attribute first name (FN), then they refer to the same patient. Therefore, the attribute blood group (BG) of $\mathsf{R_1}$ corresponds to the attribute blood type (BT) of $\mathsf{R_2}$, and the remaining common attributes birthdate (BD) and sex are equal.

Since these are only few examples of RFDS, it is not easy to understand whether there is an RFD in the literature that is suitable to solve a specific problem, or if a new one is needed. Moreover, when there is more than one suitable RFD, one would like to have some criteria to choose the most effective one. Thus, there is the need to devise a systematic way to compare the features of different RFDS, and to select the one that is most suitable for a specific problem. These issues are the focus of the next section.

3 CLASSIFICATION OF RFDS

Since the focus of this survey is on relaxed FDS, it is important to categorize each RFD based on its underlying relaxation criteria. To this end, as said in the previous section, the type of tuple comparison used on the LHS and RHS of the RFD is one of such criteria, and we will refer to it by using the term *attribute comparison*. Moreover, another relaxation criteria is based on the satisfiability property, admitting the possibility that an RFD might not be satisfied for a subset of tuples. In the rest of the paper we will use the term *extent* to refer to this relaxation criteria, which indicates whether an RFD is satisfied by a *subset* or *all* the tuples.

The *attribute comparison* and *extent* relaxation criteria are further detailed in the diagram shown in Fig. 2, which also

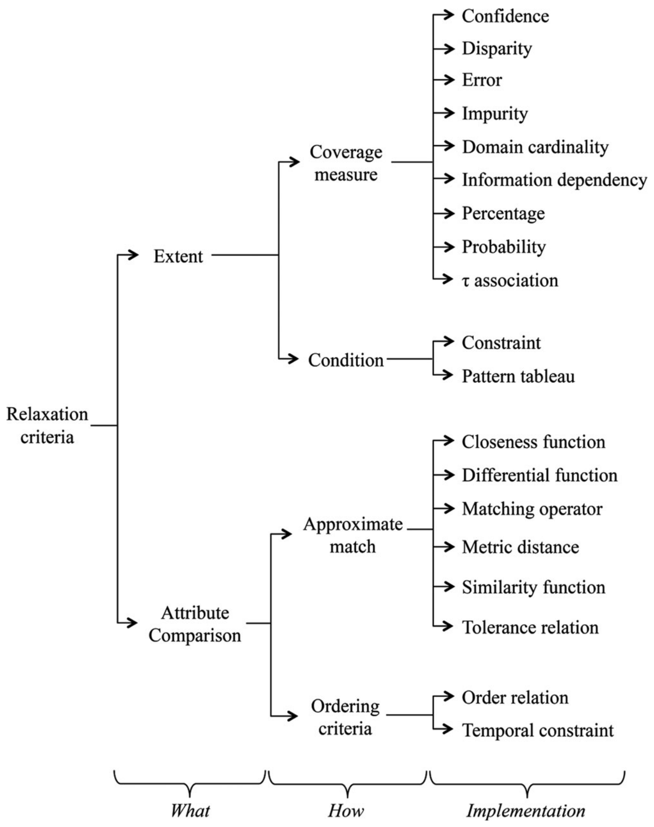

Fig. 2. Characteristics of relaxation criteria for RFDS.

provides several parameters to classify the surveyed RFDS. In particular, for the RFDS relaxing on the *attribute comparison* two main categories of paradigms are considered: *approximate match* and *ordering criteria*. The former is used to quantify the similarity or the diversity of attribute values, whereas the second one compares them based on a given order relation. On the other hand, the RFDS relaxing on the *extent* hold for "almost" all tuples or for a subset of them. In the first case, a *coverage measure* should be specified to quantify the degree of satisfiability of the RFD, whereas in the second case a condition should be specified to identify the subset of tuples.

In the following, we recall some general definitions that will let us introduce a general semantics for identifying classes of RFDS and establishing relationships among them.

Database concepts. A relational database schema $\mathcal{R}$ is a collection of relation schemas $(R_1, \ldots, R_n)$, where each R_i is defined over a fixed set of attributes $attr(R_i)$. Each attribute A_k has associated a domain $dom(A_k)$, which can be finite or infinite. A relation instance (or simply a relation) r_i of R_i is a set of tuples such that for each attribute $A_k \in attr(R_i)$, $t[A_k] \in dom(A_k)\ \forall\ t \in r_i$. A database instance r of $\mathcal{R}$ is a collection of relations $(r_1, \ldots, r_n)$, where r_i is a relation instance of R_i, for $i \in [1, n]$.

An XML database can be defined as a forest of rooted, ordered, and labeled trees. Each XML tree T is composed of a finite set of nodes V having associated either an element label (tag), an attribute name, or an element value. Edges in T represent either element-subelement, element-attribute, or element-value relationships. The elements, the attributes, the data types, and their relationships are specified within an XML schema definition (XSD). Moreover, a path p in T is a sequence of nodes $(l_1, \ldots, l_n)$, where l_1 is the root of T. A set P of paths is said to be *downward closed* if for any path $p \in P$, if p_1 is a strict prefix of p then $p_1 \in P$ [17].

Constraint. A constraint is defined as a restriction on some values. In the context of RFDs, it expresses the "closeness" of two values on a specific domain, and it is represented by a function ϕ. In particular, given two attributes A and B on a domain D, $\phi(A, B)$ evaluates the similarity/distance of A and B after a possible update of their values. As an example, ϕ can be defined in terms of a similarity metric $\approx$, like for instance the edit or Jaro distances (see [18] for a survey), such that $a \approx b$ is true if a and b are "close" enough w.r.t. a predefined threshold. Alternatively, ϕ could represent the matching operator $\rightleftharpoons$ [12], which first updates a and b, and then compares the derived values through the equality operator.

Coverage Measure. Given a database instance r of $\mathcal{R}$ and two sets of attributes $X, Y \subseteq attr(\mathcal{R})$ representing the LHS and RHS, resp., of an RFD φ, a coverage measure Ψ on φ quantifies the satisfiability degree of φ on r, and can be defined as a function $\Psi : dom(X) \times dom(Y) \rightarrow \mathbb{R}$ measuring the amount of tuples in r violating or satisfying φ. As an example, the *confidence measure* evaluates the maximum number of tuples $r_1 \subseteq r$ such that φ holds in r_1 [19].

RFD *syntax*. Consider a relational schema $\mathcal{R}$ defined over a fixed set of attributes, and $R_1 = (A_1, \ldots, A_k)$ and $R_2 = (B_1, \ldots, B_m)$ two relation schemas of $\mathcal{R}$. An RFD φ on $\mathcal{R}$ is denoted by

$$\mathbb{D}_{c_1} \times \mathbb{D}_{c_2} : (X_1, X_2)_{\Phi_1} \xrightarrow{\Psi \geq \epsilon} (Y_1, Y_2)_{\Phi_2}, \quad (1)$$

where

- $\mathbb{D}_{c_1} \times \mathbb{D}_{c_2} = \{(t_1, t_2) \in dom(R_1) \times dom(R_2) | (\bigwedge_{i=1}^{k} c_{1_i} (t_1[A_i])) \bigwedge (\bigwedge_{j=1}^{m} c_{2_j}(t_2[B_j]))\}$ where $c_1 = (c_{1_1}, \ldots, c_{1_k})$ and $c_2 = (c_{2_1}, \ldots, c_{2_m})$, with c_{1_i} and c_{2_j} predicates on $dom(A_i)$ and $dom(B_j)$, respectively, that filter the tuples on which φ applies;
- $X_1, Y_1 \subseteq attr(R_1)$, and $X_2, Y_2 \subseteq attr(R_2)$, with $X_1 \cap Y_1 = \emptyset$ and $X_2 \cap Y_2 = \emptyset$;
- Φ_1 (Φ_2, resp.) is a set of constraints $\phi[X_1, X_2]$ ($\phi[Y_1, Y_2]$, resp.) on attributes X_1 and X_2 (Y_1 and Y_2, resp.). For any pair of tuples $(t_1, t_2) \in \mathbb{D}_{c_1} \times \mathbb{D}_{c_2}$, the constraint $\phi[X_1, X_2]$ ($\phi[Y_1, Y_2]$, resp.) indicates true, if the distance/similarity of t_1 and t_2 on attributes X_1 and X_2 (Y_1 and Y_2, resp.) agrees with the constraint specified by $\phi[X_1, X_2]$ ($\phi[Y_1, Y_2]$, resp.).
- Ψ is a coverage measure defined on $\mathbb{D}_{c_1} \times \mathbb{D}_{c_2}$.
- ϵ is a threshold indicating the lower bound (or upper bound in case the comparison operator is $\leq$) for the result of the coverage measure.

RFD *semantics.* Given $r_1 \subseteq \mathbb{D}_{c_1}$ and $r_2 \subseteq \mathbb{D}_{c_2}$ two relation instances on (R_1, R_2), (r_1, r_2) *satisfies* the RFD φ, denoted by $(r_1, r_2) \models \varphi$, if and only if: $\forall$ $(t_1, t_2) \in (r_1, r_2)$, if $\phi[X_1, X_2]$ indicates true for each constraint $\phi \in \Phi_1$, then *almost always* $\phi[Y_1, Y_2]$ indicates true for each constraint $\phi \in \Phi_2$. Here, *almost always* means that $\Psi(\pi_{X_1}(r_1)\pi_{X_2}(r_2), \pi_{Y_1}(r_1)\pi_{Y_2}(r_2)) \geq \epsilon$.

In other words, if $t_1[X_1]$ and $t_2[X_2]$ agree with the constraints specified by Φ_1, then $t_1[Y_1]$ and $t_2[Y_2]$ agree with the constraints specified by Φ_2 with a degree of certainty (measured by Ψ) greater than ϵ.

For RFDs defined on single relation schemas (i.e., $R_1 = R_2$), if $X_1 = X_2$ and $Y_1 = Y_2$, then we will simplify the RFD notation given in (1) by using $\mathbb{D}_c$, X, and Y to refer to the Cartesian product $\mathbb{D}_{c_1} \times \mathbb{D}_{c_2}$, and to the pairs (X, X) and (Y, Y), respectively. Moreover, if the RFD has to be satisfied by all tuples in r, then the symbol $\Psi_{err(0)}$ is shown on the arrow. Such coverage measure corresponds to the expression $\psi(X, Y) = 0$, where $\psi(X, Y)$ measures the number of tuples violating the RFD. As an example, the canonical FD can also be written as:

$$\mathbb{D}_{\text{TRUE}} : X_{\text{EQ}} \xrightarrow{\Psi_{err(0)}} Y_{\text{EQ}}, \quad (2)$$

where TRUE is a sequence of tautologies, hence $\mathbb{D}_{\text{TRUE}} = dom(R_1)$, whereas EQ is the equality constraint.

If the database is modeled through XML instead of the relational data model, then the definition given in (1) applies to a set of trees DT complying with a given XSD, by considering $\mathbb{D}_c$ as the subset of DT satisfying predicates in c, X and Y as sets of paths. Given this, an exact functional dependency for XML (XFD) can be written as in (2).

In the following sections we will use the proposed semantics to describe the surveyed RFDs. In particular, we will group RFDs based on the two relaxation criteria described above. Thus, in Section 4 we present RFDs relaxing on the *extent* only, in Section 5 those relaxing on the *attribute comparison* only, and finally, in Section 6 those relaxing on both. Moreover, Table 3 provides the abbreviations used in such sections to refer to the surveyed RFDs.

4 FDs Relaxing on the Extent

As said above, relaxing on the *extent* parameter means that the FD holds on "almost" all tuples or on a subset of them. In fact, FDs might not hold for some tuples due to errors, missing values, or different data formats, but also to the possibility for some application domains to admit outliers.

In this section we review 11 RFDs falling in this class. They mainly differ on the method used to specify the subset of tuples for which the RFD is satisfied. In particular, those using a *coverage measure* can be defined as

$$\mathbb{D}_{\text{TRUE}} : X_{\text{EQ}} \xrightarrow{\Psi(X,Y)} Y_{\text{EQ}},$$

where Ψ represents the coverage measure. Six out of the 11 reviewed RFDs fall in this category; three of them use the probability, one uses the domain cardinality, while the other two are based on the error and the impurity measures, respectively. On the other hand, the remaining five RFDs relying on a condition to identify the subset of tuples can be defined as

$$\mathbb{D}_c : X_{\text{EQ}} \xrightarrow{\Psi_{err(0)}} Y_{\text{EQ}},$$

where $\mathbb{D}_c$ is the subset identified by the sequence of predicates in c. Two of them use constraints as predicates, whereas the other three use pattern tableaux.

The 11 RFDs reviewed in this section have been defined for many different application domains, with particular

TABLE 3
A Summary of RFD Abbreviations

RFD abbrev.	RFD name
ACOD	Approximate comparable dependency
ADD	Approximate differential dependency
AFD	Approximate functional dependency
COD	Comparable dependency
CFD	Conditional functional dependency
CFD^p	CFD with built-in predicates
CFD^c	CFD with cardinality constraints and synonym rules
CMD	Conditional matching dependency
CSD	Conditional sequential dependency
CD	Constrained functional dependency
DD	Differential dependency
eCFD	Extended conditional functional dependency
FFD	Fuzzy functional dependency
MD	Matching dependency
MFD	Metric functional dependency
ND	Neighborhood dependency
NuD	Numerical dependency
OD	Order dependency
OD_K	OD satisfied within bound k
OD_{EA}	OD satisfied almost everywhere
OFD	Ordered functional dependency
PD	Partial determination
POD	Polarized order dependencies
preFD	Preference functional dependency
PAC	Probabilistic approximate constraint
pFD	Probabilistic functional dependency
PuD	Purity dependency
RUD	Roll-up dependency
SD	Sequential dependency
SFD	Similarity functional dependency
soft FD	Soft functional dependency
TD	Trend dependency
TMFD	Type-M functional dependency
XCFD	XML conditional functional dependency
$\sigma\theta$XFD	XML FD with σ and θ approximation

emphasis on data quality and query related problems, such as query answering, rewriting, and optimization. Ten of them have been proposed for the relational data model, whereas one is defined for XML databases, which are becoming popular due to the necessity of modeling semi-structured data, like those available on the web.

4.1 Approximate Functional Dependencies (AFDs)

AFDs are FDs holding on almost every tuple [16], [19]. In order to quantize how an AFD "almost" holds, several measures have been proposed, such as the g_3 error measure [20], which corresponds to the minimum number of tuples to be removed from a relation instance r in order for an FD $X \rightarrow Y$ to hold. The tuples violating the FD are quantized by means of the error defined as

$$\psi(X,Y) = \frac{min\{|r_1| \,|\, r_1 \subseteq r \text{ and } X \rightarrow Y \text{ holds in } r \backslash r_1\}}{|r|}.$$

Formally, given an error threshold ϵ, $0 \leq \epsilon \leq 1$, an AFD can be written as

$$\mathbb{D}_{\text{TRUE}} : X_{\text{EQ}} \xrightarrow{\psi(X,Y) \leq \epsilon} Y_{\text{EQ}}.$$

As an example, in the medical database of Fig. 1, it is unlikely to have two patients with the same name that have been admitted to the hospital. Thus, except for few homonymy cases, the Name of the Patient should imply the BloodType. Thus, the following AFD might hold:

$$\mathbb{D}_{\text{TRUE}} : \text{Name}_{\text{EQ}} \xrightarrow{\psi(X,Y) \leq 0.02} \text{BloodType}_{\text{EQ}}.$$

4.2 Purity Dependencies (PuDs)

Purity dependencies generalize canonical FDs based on the notion of impurity measure [21]. The latter is primarily defined as a measure of a subset L of a set S with respect to a partition Π_S of it. In particular, if L is completely contained in a block of Π_S, then its impurity is 0. Thus, an impure set intersects more blocks in Π_S. This concept can be generalized to measure the impurity between two partitions Π'_S and Π''_S of S, which is 0 when each block of Π'_S is included in one block of Π''_S, and it increases when the blocks of Π'_S intersect more blocks of Π''_S.

Since the attribute values of a database relation induce a partition on the set of tuples, we can observe that a canonical FD $X \rightarrow Y$ induces two partitions on attribute sets X and Y, respectively, without impurity. On the other hand, when the impurity induced by the attribute sets X and Y is greater than 0, but below a given threshold, a PuD can be used.

Formally, let r be a database instance, X and Y be two sets of attributes, π_X and π_Y be two partitions of the set of tuples of r induced by the values of X and Y in r, respectively (i.e., π_X and π_Y correspond to the groups returned by SQL clauses *group by X* and *group by Y*). A PuD is defined as

$$\mathbb{D}_{\text{TRUE}} : X_{\text{EQ}} \xrightarrow{\theta(\pi_X, \pi_Y) \leq \epsilon} Y_{\text{EQ}},$$

where θ is a concave and subadditive function that computes the largest impurity measure on the blocks in π_X relative to π_Y.

As an example, in a real size instance of the relation Medicine of Fig. 1, the following PuD might hold:

$$\mathbb{D}_{\text{TRUE}} : \text{ActiveIngredient}_{\text{EQ}} \xrightarrow{\theta(\pi_{\text{ActiveIngredient}}, \pi_{\text{Rx}}) \leq 0.09} \text{Rx}_{\text{EQ}},$$

where Rx indicates whether a medical prescription is mandatory. Notice that, even if the previous PuD would reduce to a canonical FD on the sample instance of Fig. 1, in a real size instance there might be medicines with the same ActiveIngredient and different Rx values, such as Oki and Ketodol.

4.3 Numerical Dependencies (NuDs)

Numerical dependencies are FDs relaxing on the extent by means of a cardinality constraint [22]. The latter specifies a constraint on the domain cardinality for a given attribute [23]. In particular, given a relation R, and $X, Y \subseteq attr(R)$, a NuD specifies that each tuple $t[X]$ is associated to at most k different tuples on Y, for some constant k.

NuDs have been mainly used in the context of database design, when vertical decomposition (projection) needs to be augmented by horizontal decomposition (splitting) [22]. Formally, a NUD on a relation R has the form

$$\mathbb{D}_{\text{TRUE}} : X_{\text{EQ}} \xrightarrow{card(X,Y) \leq k} Y_{\text{EQ}},$$

where $card(X,Y) = |\pi_Y(\sigma_{(X=t[X])}(r))|$. As an example, in the medical database of Fig. 1, if a single ward in the hospital can have at most 10 assigned rooms, then the following NUD holds on the relation ClinicalRecords:

$$\mathbb{D}_{\text{TRUE}} : \text{CheckinWard}_{\text{EQ}} \xrightarrow{card(\text{CheckinWard}, \#\text{Room}) \leq 10} \#\text{Room}_{\text{EQ}}.$$

4.4 Probability-Based Functional Dependencies

Among the FDs holding on almost every tuple, it is worth mentioning probability-based FDs, which exploit the probability to approximate on the *extent*. In this section, we survey three probability-based RFDS, namely *partial determinations* (PDS), *soft functional dependencies* (soft FDs), and *probabilistic functional dependencies* (pFDS). In particular, although their definitions appear to be equivalent, they have been introduced in different periods, for different goals, and use different algorithms for evaluation and discovery purposes.

Partial determinations have been introduced in the late '80s, aiming to discover functional dependencies from data even in case of exceptions [24], [25]. Formally, a PD is an expression of the form

$$\mathbb{D}_{\text{TRUE}} : X_{\text{EQ}} \xrightarrow{\varrho(X,Y) \geq 1-\epsilon} Y_{\text{EQ}}, \quad (3)$$

where $\varrho(X,Y)$ represents the probability that two randomly chosen tuples have the same values of Y, provided they have the same values of X [24].

PDS generalize both the concepts of canonical FD and association rule. Nevertheless, their authors show that the measures used for evaluating the goodness of association rules, namely support and confidence, are not suitable for evaluating partial determinations. Thus, they define a compression-based measure, relying on the Minimum Description Length principle [25], [26]. Such a measure has also been used to estimate the data compression degree achievable by using a given PD.

With respect to PDS, *soft functional dependencies* have been defined several years later, in the mid noughties [27]. They have been mainly used for selectivity estimation in query optimization [27], and for recommending attributes on which to construct secondary indices [28]. Although a soft FD obeys to the same formal definition of (3), the values of the function $\varrho(X,Y)$ are computed on a database instance by means of a different discovery algorithm.

Probabilistic functional dependencies have been defined in the late noughties to tackle problems related to data integration processes from multiple data sources [29], including web data sources. In this context, it might be necessary to generate FDs from statistics over the data, due to their uncertainty and poor quality. However, like PDS and soft FDs, also pFDS are based on the notion of probability and obey to the same formal definition of (3). Thus, except for the use of lower values for ϵ, due to the different characteristics of the application domain they target, there is no difference between pFDS and PDS.

As an example, in the relation Medicine of Fig. 1, the canonical FD Name $\rightarrow$ Producer might not always be satisfied due to the presence of dirty data, like in the case of medicines with similar names (e.g., Daflon versus Deflan or Lanoxin versus Laroxyl). Thus, the following RFD might be inferred from data, by means of one of the discovery algorithms provided by the probabilistic-based FDS described above:

$$\mathbb{D}_{\text{TRUE}} : \text{Name}_{\text{EQ}} \xrightarrow{\varrho(\text{Name}, \text{Producer}) \geq 0.97} \text{Producer}_{\text{EQ}}.$$

4.5 Constrained Dependencies (CDS)

Constrained functional dependencies specify the subset of tuples on which an FD is satisfied by means of a constraint [30]. They have been mainly used in the context of constraint logic programming (CLP), aiming to perform subsumption analysis and performance optimization in top-down CLP systems.

Formally, a CD on a relation R and class of constraints $\mathcal{L}$, has the form

$$\mathbb{D}_c : X_{\text{EQ}} \xrightarrow{\Psi_{err(0)}} Y_{\text{EQ}},$$

where $X, Y \subseteq attr(R)$, $\mathbb{D}_c \subseteq dom(R)$ represents the tuples satisfying the constraint $c \in \mathcal{L}$ with variables from $attr(R)$.

As an example, in the medical database of Fig. 1, suppose that only for Pediatricians the Salary depends on the work seniority (Experience), the following CD holds on the relation Doctor:

$$\mathbb{D}_c : \text{Experience}_{\text{EQ}} \xrightarrow{\Psi_{err(0)}} \text{Salary}_{\text{EQ}},$$

where $\mathbb{D}_c = \{t \in dom(\text{Doctor}) | t[\text{Specialization}] = \text{'Pediatrician'}\}$.

4.6 Conditional Functional Dependencies (CFDS)

Conditional functional dependencies were first proposed for data cleaning purposes [11]. Similar to CDS, they use conditions to specify the subset of tuples on which a dependency holds. However, conditions are less general than CD constraints, since they only enable the specification of constraints based on the equality operator.

Formally, a CFD over a relation R is defined as

$$\mathbb{D}_{T_r} : X_{\text{EQ}} \xrightarrow{\Psi_{err(0)}} Y_{\text{EQ}},$$

where $\mathbb{D}_{T_r}$ is the domain of values satisfying a pattern tableau T_r with attributes in X and Y. In particular, for each tuple $t_p \in T_r$ and each attribute $A \in X \cup Y$, $t_p[A]$ is either a constant 'a' in $dom(A)$, or an unnamed variable '_' taking values from $dom(A)$. Intuitively, the pattern tableau T_r extracts a subset of tuples from $dom(R)$ by enforcing the binding of semantically related data values.

Referring to the example given in Section 4.5, also a CFD holds on the relation Doctor by using the following tableau

Specialization	Experience	Salary
Pediatrician	_	_

Several studies have been made on CFDs. Cong et al. used CFDs for finding repairs upon database updates in data cleaning contexts [31]. They compute repairs satisfying a given set of CFDs, also providing heuristic-based algorithms. Fan investigated the problem of CFD propagation [32], which consists in determining whether a set of CFDs are still valid upon the definition of views on a given data source. Moreover, algorithms for estimating the confidence of CFDs have also been proposed [33].

Several extensions of CFDs have also been produced. Bravo et al. proposed *extended conditional functional dependencies* (eCFDs) extending the conditions with disjunction and inequality [34]. Golab et al. defined a range tableau for CFDs [35], where each value is a range. Finally, Chen et al. introduced CFD^{p}s, which enable the specification of patterns of data values with $<$, $\leq$, $>$, $\geq$, and $\neq$ predicates [36].

4.7 CFDs for XML Databases

XML conditional FDs (XCFDs) have been defined to improve the capability of previously defined XFDs in detecting semantic constraints within XML data [37], and hence data inconsistencies. This improvement is achieved by defining XFDs capable of capturing conditional semantics that apply to some fragments rather than the entire XML tree.

Formally, given an XML database DT complying with an XSD ST, an XCFD holding on DT is defined as:

$$\mathbb{D}_c : X_{\text{EQ}} \xrightarrow{\Psi_{err(0)}} Y_{\text{EQ}},$$

where $\mathbb{D}_c = \{T \mid T$ is a tree complying with ST and satisfying condition $c\}$.

As an example, a hospital might keep an XML database for the emergency room to manage priority based emergency requests. Among the data stored within such a database, we can imagine a priority code depending on the urgency of the request, and a percentage of charge the patient should pay. Since the latter depends on the health insurance of the patient, we might define generic XFDs such as "*all patients having the same health insurance company and the same contract pay the same percentage of charge*". However, an XCFD would be necessary to express constraints applying to XML subtrees, such as "*all the patients having a premium contract with* DoubleCross *health insurance company pay no charge*".

5 FDs Relaxing on the Attribute Comparison

Relaxing FDs on the *attribute comparison* means using approximate matching paradigms to compare the attribute values on the LHS and the implied attribute values on the RHS. This yields RFDs capable of capturing semantic relations between groups of values that appear to be "similar" rather than identical.

In this section we review 16 RFDs falling in this class, which have the following structure:

$$\mathbb{D}_{\text{TRUE}} : X_{\Phi_1} \xrightarrow{\Psi_{err(0)}} Y_{\Phi_2},$$

where Φ_1 and Φ_2 are set of constraints. In particular, eight of them use approximate matching, whereas the remaining ones rely on ordering criteria.

Most of the RFDs revised in this section have been defined for solving data quality problems. Other application domains include query optimization and knowledge discovery problems. Moreover, the majority of them have been defined for the relational data model, but the remaining ones have been proposed for several additional data models, such as the dataspace, the ordered, the fuzzy, and the temporal.

5.1 Metric Functional Dependencies (MFDs)

When integrating data from various sources, it might happen that small variations in the data format and/or in the interpretation cause canonical FDs to be violated, even if an intrinsic violation of semantics does not occur. Examples include different formats for addresses, phone numbers, dates, and currencies. Koudas et al. defined *metric functional dependencies* [38], which generalize the canonical FD by tolerating small differences (controlled by a metric) in the values of the consequent attribute of an FD.

Formally, let $\phi : dom(Y) \times dom(Y) \rightarrow \mathbb{R}$ be a metric defined on the domain of attribute Y. Thus, ϕ is symmetric, and satisfies the triangle inequality as well as the identity of indiscernibles ($\phi(a,b) = 0 \Leftrightarrow a = b$). Let $\Delta_\phi(V)$ be the diameter of a set of points V in a metric space, defined as the maximum distance between any pair of points: $\Delta_\phi(V) = max_{a,b \in V}\phi(a,b)$. Given a relation r defined over a relation schema R, a subset of tuples $s \subseteq r$, two subsets of attributes $X, Y \subseteq attr(R)$, a metric ϕ over Y, and a parameter $\epsilon \geq 0$, an MFD is defined as

$$\mathbb{D}_{\text{TRUE}} : X_{\text{EQ}} \xrightarrow{\Psi_{err(0)}} Y_{max_{s\in\pi_X}\Delta_\phi(s[Y])\leq\epsilon},$$

where π_X represents the partition of r with respect to the values of the attribute X, and $s[Y]$ represents the projection of s onto the attributes in Y. In other words, the MFD states that if a group of tuples share a common value for the attribute X, then their values on the attributes of Y differ no more than ϵ.

As an example, in the medical database of Fig. 1, if the Salary gap of doctors with equal Specialization and Experience is below \$5,000, the following MFD holds on the relation Doctor:

$$\mathbb{D}_{\text{TRUE}} : (\text{Specialization,Experience})_{\text{EQ}} \xrightarrow{\Psi_{err(0)}} \text{Salary}_{max_{s\in\pi_X}\Delta_\phi(s[Y])\leq 5000}.$$

5.2 Neighborhood Dependencies (NDs)

Neighborhood dependencies have been introduced to express regularities within data [39]. Their definition is based on the association of a closeness function θ_A to each attribute A, which returns a number between 0 and 1 to express how two attribute values are close. Then, the NDs exploit the concept of neighborhood predicate, which maps each attribute A to a threshold α for its closeness function, denoted by A^{α}. Thus, given two sets of attributes $X, Y \subseteq attr(R)$, an ND is denoted as:

$$\mathbb{D}_{\text{TRUE}} : X_{(\theta_{A_1} \geq \alpha_1) \wedge .. \wedge (\theta_{A_n} \geq \alpha_n)} \xrightarrow{\Psi_{err(0)}} Y_{(\theta_{B_1} \geq \beta_1) \wedge .. \wedge (\theta_{B_m} \geq \beta_m)}$$

and it is said to be satisfied on a relation instance r of R if, whenever the closeness function θ_A satisfies the threshold α for each $A^{\alpha} \in X$, then the closeness function θ_B satisfies the threshold β for each $B^{\beta} \in Y$.

As an example, the ND

$$\mathbb{D}_{\text{TRUE}} : (\text{Diagnosis, Age})_{(\theta_{\text{Diagnosis}} \geq 0.85) \wedge (\theta_{\text{Age}} \geq 0.8)} \xrightarrow{\Psi_{err(0)}} \text{CheckinWard}_{\theta_{\text{CheckinWard}} \geq 0.9}$$

expresses the hypothesis that patients with similar Age and Diagnosis are admitted to the same CheckinWard.

5.3 Fuzzy Functional Dependencies (FFDS)

The fuzzy relational model has been introduced to represent real word situations in which the data are ambiguous or imprecise.

A fuzzy relation schema R is a finite set of attributes $\{A_1, \ldots, A_n\}$, each associated to a domain $dom(A_i)$ that can be either a fuzzy set or a set of fuzzy sets. A fuzzy relation r on R is a fuzzy subset of the Cartesian product $dom(A_1) \times \cdots \times dom(A_n)$.

In order to capture important semantic relationships in fuzzy databases, FFDs have been defined [5], [15], [40]. They extend canonical FDs by replacing the equality comparison on domain values with "approximately equal", "more or less equal", and so forth.

Formally, let us consider two sets of attributes $X, Y \subseteq attr(R)$, an FFD

$$\mathbb{D}_{\text{TRUE}} : X_{\mu_{eq} = \alpha} \xrightarrow{\Psi_{err(0)}} Y_{\mu_{eq} \geq \alpha}$$

holds on a fuzzy relation r of R, if for any pair of tuples t_1 and t_2 of r (i.e., those for which the fuzzy membership function $\mu_r(t_i) > 0$, for $i = 1, 2$), we have

$$\mu_{eq}(t_1[X],\ t_2[X]) \leq \mu_{eq}(t_1[Y],\ t_2[Y]),$$

where μ_{eq} is a fuzzy resemblance relation. The latter should be appropriately selected during the database creation, in order to capture the meaning of equality/approximate equality of the domain values from the designer point of view.

An example of FFD for the relation ClinicalRecord in Fig. 1 has been given in Section 2.

5.4 Similarity Functional Dependencies (SFDS)

Analogously to PUDs, *similarity functional dependencies* are defined in terms of partitions [41], but they relax on the comparison rather than the *extent*. They provide a sound formal characterization of the concept of similarity between pairs of tuples, based on lattice theory and formal concept analysis. In particular, starting from a definition of FD based on pattern structures [42], they derive a new RFD by introducing similarity comparisons between pattern structures, by means of tolerance relations. The latter is a relation on a set satisfying the reflexivity and the symmetry properties, but not the transitivity one.

A tolerance relation θ induces blocks of tolerance, representing subsets in which θ holds between any pairs of elements, but the property is lost upon adding any external element (maximality). For an attribute A of a relation R, a tolerance relation θ_A is defined as: $t_i \theta_A t_j \Leftrightarrow |t_i(A) - t_j(A)| \leq \epsilon$, with t_i, t_j tuples of an instance r of R. Analogously, for a set of attributes X the tolerance relation θ_X is defined as: $t_i \theta_X t_j \Leftrightarrow \forall A \in X\ t_i \theta_A t_j$. Thus, an SFD

$$\mathbb{D}_{\text{TRUE}} : X_{\theta_X} \xrightarrow{\Psi_{err(0)}} Y_{\theta_Y}$$

holds on a relation instance r if and only if $\forall t_i, t_j \in r : t_i \theta_X t_j \Rightarrow t_i \theta_Y t_j$.

As an example, for the medical database in Fig. 1, the following SFDs might hold:

$$\mathbb{D}_{\text{TRUE}} : \text{Salary}_{\theta_{\text{Salary}}} \xrightarrow{\Psi_{err(0)}} \text{Tax}_{\theta_{\text{Tax}}},$$

where θ_{Tax} depends on the Tax rates of a country, and on the maximum percentage gap between two adjacent tax levels.

5.5 Type-M Functional Dependencies (TMFDS)

The *type-M functional dependency* represents a type of RFD defined for multimedia data. It is parameterized upon distance functions used for comparing multimedia attributes [8].

In order to evaluate the similarity between the multimedia objects referred within a pair of tuples, the TMFD definition introduces the *tuple distance function* aiming to summarize the results of several distance functions computed on single tuple values. More formally, let $R(A_1, \ldots, A_n)$ be a relation schema of a multimedia database, r be a database instance of R, and t_1 and t_2 be two tuples of r, then the tuple distance function χ can be defined as

$$\chi(t_1, t_2) = \theta(\phi_1(a_1, b_1), \ \ldots, \ \phi_n(a_n, b_n)),$$

where each ϕ_i is a distance function defined on $dom(A_i)$, whereas θ represents an aggregation function $\theta : [0,1]^n \rightarrow [0,1]$.

Hence, two tuples t_1 and t_2 are similar within a threshold ϵ, denoted by $t_1 \cong_{(\chi,\epsilon)} t_2$, if and only if $\chi(t_1, t_2) \leq \epsilon$.

If $X, Y \subseteq attr(R)$, a TMFD

$$\mathbb{D}_{\text{TRUE}} : X_{\chi_1 \leq \epsilon'} \xrightarrow{\Psi_{err(0)}} Y_{\chi_2 \leq \epsilon''}$$

holds on R if and only if for any two tuples t_1 and t_2 in r, having $t_1[X] \cong_{(\chi_1,\epsilon')} t_2[X]$, then $t_2[Y] \cong_{(\chi_2,\epsilon'')} t_2[Y]$, where χ_1 and χ_2 belong to the set of distance functions defined on X and Y, respectively, whereas ϵ' and $\epsilon'' \in [0,1]$ are thresholds.

An example of TMFD for the relation Clinical Record in Fig. 1 has been introduced in Section 2.

5.6 Matching Dependencies (MDS)

Matching dependencies are mainly used in the context of record matching [12], [32]. The latter refers to the problem

Insurance					
#Policy	Insured	Bdate	Post	Premium	Balance
35677651	Andrew White	03-14-1953	987 Jefferson, NV	\$2,400	\$4500
35677712	M. Brown	08-31-1930	55 Fifth AV, NV	\$2,900	\$250
35677754	B. Gregor	12-21-1970	100 Canal Street, NJ	\$1,000	\$2430
35677788	J. Smith	11-01-2005	320 Delawere AV, CA	\$750	\$150
35677832	Erik Ford	10-19-1929	774 Vermont Street, WA	\$3,220	\$100
...					

Fig. 3. Sample relation from an health Insurance DB.

of identifying records representing the same real-world entity, even if they differ on some attribute values, due to errors or different representation formats.

Formally, an MD for a pair of relations (R_1, R_2) is defined as follows:

$$\mathbb{D}_1 \times \mathbb{D}_2 : (X_1, X_2)_{\approx} \xrightarrow{\Psi_{err(0)}} (Y_1, Y_2)_{\rightleftharpoons},$$

where (1) $X_1 = A_1, \ldots, A_n, X_2 = B_1, \ldots, B_n, Y_1 = E_1, \ldots,$ E_m, $Y_2 = F_1, \ldots, F_m$, $\mathbb{D}_1 = dom(R_1)$, $\mathbb{D}_2 = dom(R_2)$, for each $j \in [1, n]$, A_j and B_j are attributes on R_1 and R_2, respectively, sharing the same domain; similarly considerations apply for E_i and F_i when $i \in [1, m]$; (2) $\approx_j$ is a *similarity predicate* defined on the domains of $R_1[A_j]$ and $R_2[B_j]$; and (3) $\rightleftharpoons$ is a *matching operator* indicating that $R_1[E_i]$ and $R_2[F_i]$ are identified via updates, i.e., $R_1[E_i]$ and $R_2[F_i]$ are updated to make them identical.

MDs are quite different from canonical FDs in that they are defined across different relations, rather than a single one, and have "dynamic" semantics to accommodate errors and different data representation formats. Indeed, the concept of dynamic semantics refers to the fact that attributes on the RHS of an MD match as a result of an update operation. Thus, while attributes on the LHS of an MD are compared on an instance r, those on the RHS are made identical on a different instance r' resulting from the updates to r. On the contrary, in the canonical FD all the attributes are matched in the same relation instance r.

As an example, let us consider the relation in Fig. 3 from a database of health insurance contracts. Clearly, the data concerning people insured are related to those of the relation Patient from the database schema of Fig. 1, but this might not appear evident, due to possibly different data formats and attribute names. In particular, the following MD holds:

$$\mathbb{D}_{\text{Patient}} \times \mathbb{D}_{\text{Insurance}} :(\{\text{Name}, \text{Birthdate}\}, \{\text{Insured}, \text{Bdate}\})_{\approx} \xrightarrow{\Psi_{err(0)}} (\text{Addr}, \text{Post})_{\rightleftharpoons}.$$

Notice that, other than matching tuples from two different relations, the MD states that upon a match between the pair of attributes Name and Birthdate of Patient, and the pair Insured and Bdate of Insurance, respectively, the instance of the relation Insurance is evolved to make the value of the attribute Post identical to that of the attribute Addr, since the matching tuples represent the same person.

In [43], Bertossi et al. face several implementation issues related to the matching of attribute values, which where left unexplored in the original formulation of MDs. In particular, they studied the application of matching functions in the presence of multiple MDs, by providing a formal and operational semantics for MD enforcement with matching functions. Moreover, they introduce a class of matching functions satisfying several axioms, yielding a lattice framework on attribute domains. Such functions produce a value semantically dominating those of the attributes to be matched. The semantic domination represents a partial order that can be lifted to tuples of values and to database instances. The authors also define the concept of *clean* database instance, providing computational mechanisms for deriving it from a given input instance, through the enforcement of MDs in a chase-like procedure.

In [44] MDs are extended to make them applicable to fuzzy attributes. They have been exploited in the context of product databases, in which they are used as rules to improve product matching.

5.7 Comparable Dependencies (CODs)

Comparable dependencies generalize the concept of RFD to the context of heterogeneous data in dataspaces [45]. They deal with situations in which the comparison is performed between attributes with different names, by means of an attribute comparison operator. They also cover the semantics of a broad class of dependencies, including FDs, MFDs, and MDs.

The comparison operator, denoted by $\leftrightarrow_{ij}$, compares two attributes A_i, A_j in a dataspace $\mathcal{S}$ according to one of the following semantics: equality operator (i.e., $A_i = A_j$), metric operator $\approx_\epsilon$ (evaluated to true if $\phi_{ij}(a_i, a_j) \leq \epsilon$, that is, the metric distance is less than a threshold ϵ), and matching operator ($A_i \rightleftharpoons A_j$). Notice that when the comparison operator corresponds to equality, COD reduces to FD.

A general comparison function is defined as $\theta(A_i, A_j)$: $[A_i \leftrightarrow_{ii} A_i, A_i \leftrightarrow_{ij} A_j, A_j \leftrightarrow_{jj} A_j]$ and specifies a constraint on comparable correspondences of values for attributes A_i or A_j. Notice that, in order to verify whether two objects t_1 and t_2 satisfy a comparison function $\theta(A_i, A_j)$, it is necessary that one of the following comparisons be true: $(t_1[A_i] \leftrightarrow_{ii} t_2[A_i]), (t_1[A_i] \leftrightarrow_{ij} t_2[A_j]), (t_2[A_i] \leftrightarrow_{ij} t_1[A_j]), (t_1[A_j] \leftrightarrow_{jj} t_2[A_j])$.

Formally, a COD with general comparison functions over a dataspace $\mathcal{S}$ can be defined as

$$\mathbb{D}_{\text{TRUE}} : (X_1, X_2)_{\theta(X_1,X_2)} \xrightarrow{\Psi_{err(0)}} (Y_1, Y_2)_{\theta(Y_1,Y_2)},$$

where $\theta(X_1, X_2) = \bigwedge \theta(A_i, A_j)$ and $\theta(Y_1, Y_2) = \bigwedge \theta(B_k, B_l)$, with $A_i \in X_1$, $A_j \in X_2$, $B_k \in Y_1$, $B_l \in Y_2$; $\theta(A_i, A_j)$ and $\theta(B_k, B_l)$ are comparison functions in the dataspace $\mathcal{S}$.

For instance, in a medical dataspace containing clinical records from several hospitals, the following COD might hold:

$$\mathbb{D}_{\text{TRUE}} : (\text{\#Room}, \text{\#Bedroom})_{\theta(\text{\#Room},\text{\#Bedroom})} \xrightarrow{\Psi_{err(0)}} (\text{Sex}, \text{Gender})_{\theta(\text{Sex},\text{Gender})}$$

since each patient room of an hospital univocally identifies the gender of the patients occupying it.

5.8 Differential Dependencies (DDS)

Song and Chen [46] introduced *differential dependencies* to express constraints on attribute value differences by means of a metric distance [18]. In particular, a *differential function* $\phi[B]$ on an attribute B specifies a constraint on the difference that tuples t_1 and t_2 have on B, and when it is satisfied it is denoted by $(t_1, t_2) \asymp \phi[B]$. The constraints are specified by operators $\{=, <, >, \leq, \geq\}$, and are associated to a threshold. As an example, given the following differential function $\phi[\mathsf{name}] = [\mathsf{name}(\leq 6)]$, we have that $(t_1, t_2) \asymp [\mathsf{name}(\leq 6)]$ if the difference between tuples t_1 and t_2 on the attribute name is less or equal than 6. The concept of differential function can also be applied to sets of attributes by means of a disjunction on the constraints defined for the single attributes.

Formally, a DD over a relation R has the form

$$\mathbb{D}_{\text{TRUE}} : X_{\phi_L} \xrightarrow{\Psi_{err(0)}} Y_{\phi_R},$$

where $X, Y \subseteq attr(R)$, ϕ_L and ϕ_R are differential functions on attribute sets X and Y, respectively. It states that for any pair of tuples for which the difference on attributes X satisfies the constraints specified by $\phi_L[X]$, then also the difference on attributes Y must satisfy the constraints specified by $\phi_R[Y]$.

As an example, to detect anomalies in the execution of medical checks with respect to the PrescriptionDate, the following DD can be used:

$$\mathbb{D}_{\text{TRUE}} : \mathsf{PrescriptionDate}_{\phi_L} \xrightarrow{\Psi_{err(0)}} \mathsf{ExecutionDate}_{\phi_R},$$

where $\phi_L[\mathsf{PrescriptionDate}] = [\mathsf{PrescriptionDate}(= 0)]$ and $\phi_R[\mathsf{ExecutionDate}] = [\mathsf{ExecutionDate}(\leq 5)]$. In particular, the DD says that medical checks prescribed on the same day should not be executed at more than five days of distance.

It is worth to notice that when the differential function on the LHS corresponds to the equality function, i.e., $\phi_L[X] : X(= 0)$, and the one on the RHS is $\phi_R[Y] : Y(\leq \alpha)$, then the DD reduces to an MFD.

5.9 Order Dependencies (ODs)

Order dependencies have been introduced to express semantic information concerning the orderings on the attribute domains of a relation [47]. Thus, their definition is based on the concept of order relation over attribute domains.

In order to formally define ODs, we need to first introduce the *marked differential functions* ($\phi_=[A]$, $\phi_\leq[A]$, $\phi_\geq[A]$) of an attribute A, which indicate the order relation holding between two tuples t_1, t_2 on A. In particular, $\phi_=[A]$ means $t_1[A] = t_2[A]$, $\phi_\leq[A]$ means $t_1[A] \leq t_2[A]$, whereas $\phi_\geq[A]$ means $t_1[A] \geq t_2[A]$. Therefore, given a relation R and two sets of attributes $X, Y \subseteq attr(R)$, an OD

$$\mathbb{D}_{\text{TRUE}} : X_{\phi_L} \xrightarrow{\Psi_{err(0)}} Y_{\phi_R}$$

is satisfied by an instance r of R if, for all tuples t_1 and t_2 in r, $(t_1, t_2) \asymp \phi_L[X]$ implies $(t_1, t_2) \asymp \phi_R[Y]$, where ϕ_L and ϕ_R are marked differential functions on the attribute sets X and Y, respectively.

As an example, in the relation ClinicalRecord in Fig. 1, the OD

$$\mathbb{D}_{\text{TRUE}} : \mathsf{Num}_{\phi_\leq} \xrightarrow{\Psi_{err(0)}} \mathsf{Date}_{\phi_\leq}$$

might hold, since it is expected that clinical records numbers are issued in a chronological order.

Finally, ODs have been generalized to introduce relaxation principles [48]. In particular, the relaxed OD is said to be satisfied within a bound $k \geq 0$ (OD$_K$, for short) if there exists a monotonically non-decreasing function $\theta : dom(X) \rightarrow dom(Y)$, such that, $\theta(t[X]) - k \leq t[Y] \leq \theta(t[X]) + k$ for each tuple t in the database instance.

5.10 Ordered Functional Dependencies (OFDs)

Ordered functional dependencies extend ODs by relaxing on the classification of domain ordering types [49], [50], enabling the possibility to capture the lexicographical ordering. The semantics of an OFD is specified through the definition of two possible extensions of the domain ordering: *pointwise-ordering* and *lexicographical ordering*.

Formally, let $Z = Z_1 \times \cdots \times Z_n$ be the Cartesian product of n ordered sets, a *pointwise-ordering* on Z, denoted by $\leq_Z^p$, indicates that given $t_1, t_2 \in Z$, then $t_1 \leq_Z^p t_2$ if for all $1 \leq i \leq n$, $t_1[i] \leq_{Z_i} t_2[i]$ holds. In other words, a *pointwise-ordering* requires that each component of a data value be greater than its predecessor, and it can be defined through a marked differential function denoted by $\phi_{\leq_Z^p}$. Instead, a *lexicographical ordering* on Z, denoted by $\leq_Z^l$, indicates that given $t_1, t_2 \in Z$, then $t_1 \leq_Z^l t_2$ if either (1) there exists a k in the range $1 \leq k \leq n$, such that $t_1[k] \leq_{Z_k} t_2[k]$, and $t_1[i] = t_2[i]$ holds for all $1 \leq i < k$, or (2) $t_1[i] = t_2[i]$ holds for all $1 \leq i \leq n$. In other words, a *lexicographical ordering* resembles the way in which the words are listed in a dictionary, and it can be defined through a marked differential function denoted by $\phi_{\leq_Z^l}$.

On the basis of these ordering semantics, two types of OFDs have been defined: OFDs *arising from pointwise-orderings* (POFDs), and OFDs *arising from lexicographical orderings* (LOFDs).

Formally, a POFD

$$\mathbb{D}_{\text{TRUE}} : X_{\phi_{\leq_X^p}} \xrightarrow{\Psi_{err(0)}} Y_{\phi_{\leq_Y^p}}$$

holds on a database instance r on R, if $\forall\ t_1, t_2 \in r$, $t_1[X] \leq_X^p t_2[X]$ entails $t_1[Y] \leq_Y^p t_2[Y]$; instead, a LOFD

$$\mathbb{D}_{\text{TRUE}} : X_{\phi_{\leq_X^l}} \xrightarrow{\Psi_{err(0)}} Y_{\phi_{\leq_Y^l}}$$

is satisfied in a database instance r on R, if for all the tuples $t_1, t_2 \in r$, $t_1[X] \leq_X^l t_2[X]$ entails $t_1[Y] \leq_Y^l t_2[Y]$.

As an example, if in the medical database of Fig. 1 there is the following semantic ordering on the attribute Role of the relation Doctor: {"Junior Dr." < "Specialized Dr." < "Senior Surgeon" < "Head Physician"}, and if the Salary of a doctor must be greater than that of other doctors with a lower Role and less Experience, according to a pointwise-ordering, then the following POFD is satisfied:

$$\mathbb{D}_{\text{TRUE}} : (\text{Role}, \text{Experience})_{\phi^{p}_{\leq\{\text{Role,Experience}\}}} \xrightarrow{\Psi_{err(0)}} \text{Salary}_{\phi^{p}_{\leq\text{Salary}}} .$$

Notice that, also a LOFD holds on the same attributes.

The application of LOFDs in the context of preference databases yields the definition of *preference functional dependency* (preFD) [51], which is a LOFD using a preference relation as a lexicographical ordering. They have been mainly used to maintain the consistency of preference semantics embedded in preference databases.

5.11 Polarized Order Dependencies (PODS)

A further extension of ODs for describing relationships among lexicographical orderings of sets of tuples are the *polarized order dependencies* [52], [53]. As opposed to LOFDs, PODs rely on the lexicographical ordering of the SQL *order-by* operator, so enabling both ascending and descending orders.

Formally, let $\phi_{\preccurlyeq PL} = [\phi_H|\phi_Z]$ be a list of marked differential functions on an attribute set $L \subseteq R$, with ϕ_H head of $\phi_{\preccurlyeq PL}$ and ϕ_Z tail of it. Given a pair of tuples t_1 and t_2 in a relation instance r on R, they are said to satisfy $\phi_{\preccurlyeq PL}[L]$ in r, denoted with $(t_1, t_2) \asymp \phi_{\preccurlyeq PL}[L]$, if one of the following three conditions holds: (1) ϕ_H is of the form $\phi_{\leq}[A]$ and $t_1[A] < t_2[A]$, (2) ϕ_H is of the form $\phi_{\geq}[A]$ and $t_1[A] > t_2[A]$, (3) ϕ_H is of the form $\phi_{=}[A]$, $(t_1[A] = t_2[A])$ and ($Z =[]$ or $(t_1, t_2) \asymp \phi_{\preccurlyeq PL}[Z]$).

Thus, given a relation R and two sets of attributes $X, Y \subseteq attr(R)$, a POD

$$\mathbb{D}_{\text{TRUE}} : X_{\phi_{\preccurlyeq PL}} \xrightarrow{\Psi_{err(0)}} Y_{\phi_{\preccurlyeq PL}}$$

is satisfied on an instance r of R, if and only if for each pair (t_1, t_2) in r, $(t_1, t_2) \asymp \phi_{\preccurlyeq PL}[X]$ implies $(t_1, t_2) \asymp \phi_{\preccurlyeq PL}[Y]$.

As an example, in the medical database of Fig. 1, the following POD holds:

$$\mathbb{D}_{\text{TRUE}} : \text{Salary}_{\phi_{\preccurlyeq PL}} \xrightarrow{\Psi_{err(0)}} (\text{Level}, \text{Tax})_{\phi_{\preccurlyeq PL}},$$

where $\phi_{\preccurlyeq PL} = [\phi_{\leq}[\text{Salary}], \phi_{\geq}[\text{Level}], \phi_{\leq}[\text{Tax}]]$. This POD says that when the Salary of medical doctors increases, the tax bracket (Level) decreases, and correspondingly, the Tax amount increases.

5.12 Sequential Dependencies (SDS)

Sequential dependencies generalize ODs by enabling the specification of additional relationships between ordered attributes, such as the range of gaps between consecutive sequence numbers. Formally, given a range of values g and a relation R, an SD on attribute sets $X, Y \subseteq attr(R)$, written as

$$\mathbb{D}_{\text{TRUE}} : X_{\phi_L} \xrightarrow{\Psi_{err(0)}} Y_{\phi_{R_g}}$$

is satisfied on an instance r of R if, for all pairs of consecutive tuples t_1 and t_2 in r, $(t_1, t_2) \asymp \phi_L[X]$ implies $(t_1, t_2) \asymp \phi_{R_g}(Y)$, where ϕ_L is a marked differential function on the attribute set X, which specifies the order relation of $dom(X)$, and ϕ_{R_g} is a differential function on the attribute set Y, which is satisfied when the distance between the values of Y are in the range g [54]. When such range cannot be determined, we have classical ODs. For instance, when $g = [0,\infty]$ the SD (saying that Y is strictly increasing with respect to X) corresponds to an OD.

SDs enable the definition of semantics that is useful both to improve the data quality, and to provide a statistical support in specific application areas like data mining. For instance, in a data mining application on the medical database in Fig. 1, it might be useful to ensure that the ExecutionDate of medical checks grows with respect to the PrescriptionDates.

5.13 Trend Dependencies (TDS)

Trend dependencies [55], [56] extend the canonical FD with a temporal dimension, and allow attributes with linearly ordered domains be compared over the time by means of an operator from the set $OP = \{<, =, >, \leq, \geq, \neq\}$.

A temporal relation can be viewed as a temporal series of "snapshots" of relations defined on the same attribute set. Therefore, temporal constraints can be controlled by checking the validity of the tuples in different time points. For example, to check that *"Salaries of doctors should never decrease"*, it is necessary to compare the doctor records at the time i and $i + 1$, for all temporal points i.

The TD definition is based on the concepts of directed attribute set (DAS) and of time accessibility relation (TAR). The former represents a total function from a set of attributes U to the set of possible operators OP, and is denoted with a Greek uppercase letter. A TAR indicates the tuples belonging to different snapshots that have to be compared. Formally, let t_1, t_2 be two tuples on U, and Φ be a DAS over $X \subset U$, the couple (t_1, t_2) is said to satisfy Φ, denoted with $\Phi^*_U(t_1, t_2)$ if and only if $t_1(A)\theta_A t_2(A), \forall A \in X$, where θ_A corresponds to $\Phi(A)$. Moreover, a TAR is a computable subset of the following set of pairs: (i, j) with $i < j$. For instance, the *Next* TAR represents the tuples of the next time-slice, i.e., $Next = \{(i, i + 1) \mid i \in \mathbb{N}\}$.

Formally, a TD on the attribute sets X and Y can be defined by a statement

$$\mathbb{D}^i_{\text{TRUE}} \times \mathbb{D}^j_{\text{TRUE}} : (X, X)_{\Theta_L} \xrightarrow{\Psi_{err(0)}} (Y, Y)_{\Theta_R},$$

where (i, j) belongs to a TAR, whereas Θ_L and Θ_R are DASs over X and Y, respectively.

As an example, to express the constraint *"an increasing* diastolic *blood pressure implies an increasing* systolic *blood pressure"* in the medical database of Fig. 1, the following TD can be used:

$$\mathbb{D}^i_{\text{TRUE}} \times \mathbb{D}^{i+1}_{\text{TRUE}} : ((\text{Patient}, \text{Diastolic}), (\text{Patient}, \text{Diastolic}))_{\Theta_L} \xrightarrow{\Psi_{err(0)}} (\text{Systolic}, \text{Systolic})_{\Theta_R},$$

where Θ_L(Patient) is "$=$", Θ_L(Diastolic) and Θ_R(Systolic) are "$<$".

5.14 Roll-Up Dependencies (RUDS)

Roll-up dependencies extend the canonical FD based on generalization hierarchies [57], which are widely used in OLAP and data mining applications. A generalization hierarchy contains several levels, on which an order relation $\preceq$ can be defined, yielding an ordered set named roll-up schema. Formally, given a set U of attributes, a set $\mathcal{L}$ of levels, and a set $\mathcal{D}$ of constants, a roll-up schema is a pair $(L, \preceq)$, where L is a finite subset of $\mathcal{L}$, and $\preceq$ is a partial order on L. Given $l_1, l_2 \in L$, $l_1 \prec l_2$ if and only if $l_1 \preceq l_2$ and $l_1 \neq l_2$.

A roll-up instance assigns an extension to each level and "instantiates" the generalization hierarchy. Formally, a roll-up instance over a roll-up schema $(L, \preceq)$ is a pair $(ext,\ \theta)$, where ext maps every $l \in L$ to a disjoint set of constants, and θ is a set of functions containing a total function $\theta_{l_1}^{l_2}$: $ext(l_1) \rightarrow ext(l_2)$, for each $l_1, l_2 \in L$ such that $l_1 \preceq l_2$.

Given the layered structure of a generalization hierarchy, a relation schema is defined as a set of attribute-level pairs, from which a "generalization" schema (*genschema*) can be built by replacing a level l with a level l', with $l \prec l'$, and/or by entirely omitting certain attributes.

Let G be a genschema of a relation schema R, two tuples t_1, t_2 over R are said to be G-equivalent, denoted with $t_1 \sim_G t_2$, if and only if, for each pair (A, l_1) of R, and for each pair (A, l_2) of G, $\theta_{l1}^{l2}(t_1(A)) = \theta_{l1}^{l2}(t_2(A))$. In other words, t_1 and t_2 become equal after rolling up their attribute values to the levels specified by G.

Formally, given X and Y genschemas of a relation schema R, a RUD

$$\mathbb{D}_{\text{TRUE}} : X_{\sim_X} \xrightarrow{\Psi_{err(0)}} Y_{\sim_Y}$$

holds on an instance r of R, if and only if for all tuples t_1, t_2 of r, if $t_1 \sim_X t_2$ then $t_1 \sim_Y t_2$.

As an example, in the medical database of Fig. 1, let us suppose that an administrator builds a report table by aggregating the attribute Hospitalization per room. Moreover, let us suppose that each ward has a fixed number of beds per room. The property that the total number of days of Hospitalization in a month does not change for rooms within the same ward can be expressed by the following RUD:

$$\mathbb{D}_{\text{TRUE}} :((\text{Date}, \textsc{Month}), (\#\text{Room}, \textsc{Ward}))_{\sim((\text{Date},\textsc{Month}),(\#\text{Room},\textsc{Ward}))}$$
$$\xrightarrow{\Psi_{err(0)}} (\text{Hospitalization}, \textsc{PerWard})_{\sim(\text{Hospitalization},\textsc{PerWard})}.$$

However, this RUD might not be satisfied throughout the hospital if there are significant differences in terms of Hospitalization among different wards.

6 FDS WITH HYBRID RELAXATION ON EXTENT AND ATTRIBUTE COMPARISON

In this section we describe hybrid RFDs, that is, FDs relaxing on both the *extent* and the *attribute comparison*. Most of them extend some of the RFDs described in Section 5, by introducing the relaxation on the *extent*, in order to capture additional semantics on the data, and suites the RFDs to new application contexts. In particular, we describe a native hybrid RFD, namely PAC [58], and extensions of the following RFDs introduced in the previous sections: CFDs, DDs, MDs, CODs, ODs, and SDs. Moreover, we describe a hybrid RFD for the XML database model, which extends the one defined in Section 4.7.

Most of the RFDs revised in this section have been defined for solving data quality problems, but their application domains also include query optimization and knowledge discovery. Moreover, the majority of them have been defined for the relational data model, but the remaining ones instantiate several additional data models, such as the dataspace, the ordered, the fuzzy, and the temporal data models.

6.1 PAC Functional Dependencies

In data management there are specific aspects for which theoretical principles are not generally applicable. For instance, in network databases specific problems might arise concerning data quality checking, e.g., missing poll, irregular poll, and repeated values. To this end, Korn et al. [58] have defined the *probabilistic approximate constraints* (PACs) to indicate the probability of correctness for a given value. They allow to manage the constraints on the data in a flexible manner, enabling the specification of the user vision in the structural properties of the data.

Formally, given a legal ordered domain $dom(A)$ of an attribute A, a domain PAC specifies that all the attribute values $a \in A$ fall within ϵ of $dom(A)$ with at least probability α, denoted as $Pr(a \in [dom(A) \pm \epsilon]) \geq \alpha$. Based on this definition, a PAC *functional dependency* can be formally defined as

$$\mathbb{D}_{\text{TRUE}} : X_{\phi_L^\beta} \xrightarrow{\varrho(X,Y) \geq \alpha} Y_{\phi_R^\epsilon},$$

where ϕ_L^β and ϕ_R^ϵ are differential functions, and $\varrho(X,Y)$ represents the probability that the dependency holds. In particular, it specifies that for each pair of tuples (t_i, t_j) of a database instance, if $(t_i, t_j) \asymp \phi_L^\beta[X]$, i.e., $|t_i[A_l] - t_j[A_l]| \leq \beta_l\ \forall A_l \in X$, then $Pr((t_i, t_j) \asymp \phi_R^\epsilon[Y]) \geq \alpha$, i.e., $Pr(|t_i[B_l] - t_j[B_l]| \leq \epsilon_l) \geq \alpha\, \forall B_l \in Y$.

Although this type of RFD has been created for a specific context, it can be applied to any type of database. As an example, for the health Insurance relation of Fig. 3, according to typical health insurance policies, we can assume that if the Birthdates of two insured are close, there is a high probability that their Premium are also close. This can be expressed through the following dependency:

$$\mathbb{D}_{\text{TRUE}} : \text{Birthdate}_{\phi_L^5} \xrightarrow{\varrho(X,Y) \geq 0.9} \text{Premium}_{\phi_R^{500}}.$$

6.2 CFDS with Cardinality Constraints and Synonym Rules

A further extension of CFDs, denoted by CFD^{c}s, has been proposed for capturing inconsistencies commonly found in real-life data [59]. In particular, this RFD relies on the concepts of *cardinality constraint*, introduced for NUDs [22] (see Section 4.3), and those of *synonym rules and patterns of semantically related values*. The latter enables the possibility of specifying abbreviations for some attribute values, or values related to them, so as to match values that are apparently different. As an example, it is possible to specify that the value "ECG" is an abbreviation of the value "Electrocardiogram".

In order to extend the equality concept, the authors defined a binary operator $\doteq$ on constant values, by using a finite binary relation SYN to capture synonymous rules upon generic values a and b. In particular, for any pair of values a and b, $a \doteq b$ if and only if (1) $\text{SYN}(a, b)$ or $a = b$, (2) $b \doteq a$, or (3) there exists a value c such that $a \doteq c$ and $b = c$. As an example, "Electrocardiogram" $\doteq$ "ECG".

Formally, a CFD^c defined on a relational schema R has the form

$$\mathbb{D}_{T_r} : X_{\text{SYN}} \xrightarrow{card_{LR}(X,Y) \leq n} Y_{\text{SYN}},$$

where SYN is the binary relation capturing synonymous rules, $card_{LR}(X,Y) = |(\pi_Y(\sigma_{X \doteq t[X]})|$ represents the number of distinct values for attribute Y that can be associated to a given value of X, and is upper bounded by the constant n; $\mathbb{D}_{T_r}$ is the domain of R satisfying the pattern tableau T_r.

As an example, in the medical database of Fig. 1, there might be specific checks for which a single doctor per day is available. Moreover, there might be checks stored with different synonymous and/or abbreviations. Thus, if for HIV and Mammography checks there is one doctor available per day, the following CFD^c holds

$$\mathbb{D}_{T_1} : (\text{Name}, \text{ExecutionDate})_{\text{SYN}} \xrightarrow{card_{LR}(\{\text{Name}, \text{ExecutionDate}\}, \text{Doctor}) \leq 1} (\text{Doctor})_{\text{SYN}},$$

where SYN consists of ("Electrocardiogram","ECG") and ("AIDS", "HIV"), T_1 is the following tableau

Name	ExecutionDate	Doctor
HIV	_	_
Mammography	_	_

It is worth to notice that the canonical FD is a special case of the CFD^c, in which n is 1, SYN is empty, and the pattern tuple consists of '_' only.

6.3 Approximate Differential Dependencies (ADDS)

As a further example of RFD extended to hold for "almost" every tuple, it is worth to mention *approximate differential dependencies* [60]. They are DDs whose specification is associated either to a g_3 error measure [20], or equivalently to a confidence measure [35], [61].

As an example, the relation Clinical Record in Fig. 1 might contain more than one record for some patients that have undergone several hospital admissions. For every checkin, a new clinical record is created and a Diet code is assigned based on the patient conditions and pathologies. It is expected that for checkin Dates closer than one year the Diet code does not change, or that changes slightly, unless the patient conditions have considerably changed. This situation can be modeled through the following ADD

$$\mathbb{D}_{\text{TRUE}} : (\text{SSN}, \text{Date})_{\phi_L} \xrightarrow{\psi(\{\text{SSN}, \text{Date}\}, \text{Diet}) \leq \epsilon} \text{Diet}_{\phi_R},$$

where $\phi_L = \text{SSN}(=0) \bigwedge \text{Date}(<365)$ and $\phi_R = \text{Diet}(\leq 1)$.

6.4 Conditional Matching Dependencies (CMDS)

Conditional matching dependencies bind MDs on a subset of tuples specified by conditions [62]. In particular, given a relation R, a CMD has the form

$$\mathbb{D}_c : X_{\approx} \xrightarrow{\Psi_{err(0)}} Y_{\rightleftharpoons},$$

where $X, Y \subseteq attr(R)$, $\approx$ and $\rightleftharpoons$ denote the corresponding similarity/matching operators on attributes of X and Y, respectively, $\mathbb{D}_c = \{t \in dom(R) \mid t[X_1] \approx x_1 \wedge \ldots \wedge t[X_n] \approx x_n \wedge t[Y_1] \rightleftharpoons y_1 \ldots t[Y_m] \rightleftharpoons y_m\}$, and represents the domain tuples satisfying the similarity and the matching conditions. The value x_i (y_j, resp.) is either a constant 'a' from $dom(X_i)$ ($dom(Y_j)$, resp.), or a virtual value $*$ in $dom(X_i)$ ($dom(Y_j)$, resp.) which is similar/identical to all the values in $dom(X_i)$ ($dom(Y_j)$, resp.). A CMD states that for any two tuples from an instance r of R, if they are similar on attributes X and their values are close to $x = (x_1, \ldots, x_n)$, then they are identical on attributes Y, and their values match $y = (y_1, \ldots, y_m)$.

As an example, the following CMD

$$\mathbb{D}_c : (\text{Name}, \text{Birthdate})_{\approx} \xrightarrow{\Psi_{err(0)}} \text{SSN}_{\rightleftharpoons}$$

with $\mathbb{D}_c = \{t \in dom(\text{Doctor}) \mid (t[\text{Name}] \approx \text{"Mary Brown"}) \bigwedge (t[\text{Birthdate}] \approx *) \bigwedge (t[\text{SSN}] \rightleftharpoons *)\}$ holds on an instance of the medical database in Fig. 1, if there are no two patients whose Name is similar to Mary Brown and whose Birthdates are close.

6.5 Approximate Comparable Dependencies (ACODS)

As for DDs, also CODs have been extended to cope with situations in which few records violate the dependency. In particular, *approximate* CODs have been defined in [45] by using the g_3 error [20], or the confidence [35], [61], to approximate the *extent* of the dependency. Thus, an ACOD with general comparison functions over a dataspace $\mathcal{S}$ is in the form:

$$\mathbb{D}_{\text{TRUE}} : (X_1, X_2)_{\theta_{(X_1,X_2)}} \xrightarrow{\Psi(X,Y) \leq \epsilon} (Y_1, Y_2)_{\theta_{(Y_1,Y_2)}},$$

where $\theta(X_1, X_2) = \bigwedge \theta(A_i, A_j)$, and $\theta(Y_1, Y_2) = \bigwedge \theta(B_k, B_l)$, with $A_i \in X_1$, $A_j \in X_2$, $B_k \in Y_1$, $B_l \in Y_2$, $\theta(A_i, A_j)$ and $\theta(B_k, B_l)$ are comparison functions in the dataspace $\mathcal{S}$, $\Psi(X,Y)$ is a measure evaluating how the dependency almost/approximately holds in a data instance, and ϵ is a threshold.

As an example, if we consider the COD mentioned in Section 5.7, it might not be satisfied for some data integrated from hospital wards using bed numbering only, instead of room numbering. Such a situation can be modeled with the following ACOD:

$$\mathbb{D}_{\text{TRUE}} : (\text{\#Room}, \text{\#Bedroom})_{\theta(\text{\#Room},\text{\#Bedroom})} \xrightarrow{\psi(X,Y) \leq 0.93} (\text{Sex}, \text{Gender})_{\theta(\text{Sex},\text{Gender})},$$

where the g_3 error measure ψ evaluates the number of tuples not obeying to a room numbering system.

6.6 ODs Satisfied Almost Everywhere

In Section 5.9 we have described an extension of ODs to include approximate comparisons within a bound k [48]. The authors provide a further extension of ODs to capture semantics applying to a subset of tuples.

Formally, given a relation R, $X, Y \subseteq attr(R)$, and $\alpha \in [0,1]$, an OD satisfied *almost everywhere* with disparity α (OD_{AE}, for short) has the form

$$\mathbb{D}_{\text{TRUE}} : X_{\phi_L} \xrightarrow{\psi'(X,Y) \leq \alpha} Y_{\phi_R},$$

where the coverage measure is defined as

$$\psi'(X,Y) = \frac{\{|r_1| \mid r_1 \subseteq r \text{ and } X_{\phi_L} \xrightarrow{\Psi_{err(0)}} Y_{\phi_R} \text{ holds in } r \backslash r_1\}}{|r|}$$

with r database instance of R.

For example, referring to the medical database of Fig. 1, we might assume that diagnostic checks are performed by following the order in which they have been prescribed by doctors. If the hospital adopts the policy to accept up to 20 percent of urgent requests to raise the priority of prescribed checks, then the following OD_{AE} holds:

$$\mathbb{D}_{\text{TRUE}} : \text{PrescriptionDate}_{\phi_{\leq}} \xrightarrow{\psi'(X,Y) \leq \frac{1}{5}} \text{ExecutionDate}_{\phi_{\leq}}.$$

6.7 Conditional Sequential Dependencies (CSDs)

Also for SDs there might be the necessary to capture their semantics in situations where they do not hold for all the tuples. Thus, in such domains we might want to capture the degree of satisfaction of an SD by accompanying the dependency with a confidence measure. In [54] the authors introduce the *conditional sequential dependency*, representing an SD valid for a subset of data satisfying a specific condition.

Formally, a CSD has the form

$$\mathbb{D}_{Tr} : X_{\phi_L} \xrightarrow{\Psi_{err(0)}} Y_{\phi_{R_g}},$$

where T_r represents the pattern tableau defining the tuples on which the dependency holds.

As an example, in the health insurance relation of Fig. 3, for insured under age of 35, the Premium grows proportionally to Age, since in this age range the order of magnitude of health costs is the same, whereas for older people it depends on the diseases that arise. For instance, if a diabetes arises, the Cost of Insulin might be considerably high, which might heavily affect the insurance Premium.

6.8 Hybrid RFDS for XML Databases

When dealing with XML databases it might not be sufficient relaxing on the *extent*, because there can be XML subtrees describing the same entity, but having different structures. Thus, in such cases it might be necessary to base the dependency detection on similarity comparisons between trees. To this end, in [63] authors define an XFD approximating both the *attribute comparison* and the *extent*. In particular, they define the $\sigma - approximation$ for evaluating structural and content similarity between trees, and the $\theta - approximation$ concerning the percentage of the whole database σ-approximately satisfying the XFD.

As an example, referring to the XML medical database mentioned in Section 4.7, the law might force hospitals to take care of people without health insurance, provided that they have a welfare card. Moreover, suppose that in these cases the law prescribes no charge for high priority emergency codes, like in case of risk to life, whereas for the remaining ones the lower the risk to life the higher the charge. If a manager wants to check the integrity of the priority code assignment process, s/he might want to use the following XFD embedding both a σ and a $\theta - approximation$: *"patients with the welfare card and with similar symptoms should have similar priority codes"*.

7 Discussion

We have surveyed RFDs, and have mainly characterized them based on the relaxation criteria, yielding three main categories of them. In what follows we compare the surveyed RFDs based on the supported data model and the potential application domains.

7.1 Comparison of RFDS

Table 4 synthesizes the main features of surveyed RFDs. As it can be noticed, the majority of RFDs relaxing on the *extent* use a coverage measure to define the bound of satisfiability. Moreover, among the most commonly used coverage measures there are the confidence, the g_3 error, and the probability. Concerning the type of *attribute comparison*, it is worth to notice that among the 35 surveyed RFDs, about 31 percent of them use the exact match, 26 percent base the comparison on an order relation, whereas most of the remaining ones use an approximate matching paradigm based on similarity or distance functions. Finally, eight RFDs use an hybrid combination of two or more of such criteria.

As expected, most of the RFDs have been defined for the relational data model (over 70 percent), mainly due to its popularity in both industries and academia. Among the other data models, we find the fuzzy, the temporal, and the ordered data models, together with a number of data models used in the context of emerging applications. Among these, we find the XML, mainly used in the context of semi-structured data, and the dataspace, mainly used in data integration and personal information management systems to deal with data heterogeneity issues.

The column *Application Domains* lists either the domains in which the RFD has been used, or those for which it is potentially suitable. This information can be considerably useful, especially if analyzed in combination with the other features examined above. In fact, when facing a specific problem involving the detection of dependencies among data, one can easily verify whether there already exists an RFD suitable for that purpose, or whether it is necessary to define a new specific one. This can also contribute to limit the proliferation of RFD definitions as experienced in the last decades, since some of them could be easily defined in terms of existing ones. Moreover, by analyzing last column (*Analyzed Problems*), it is also possible to analyze additional

TABLE 4
Main Features of Surveyed RFDs

RFD Name	Extent	Attribute comparison	Supported Data Model	Application Domains	Analyzed Problems*
AFD [19]	g_3 error [19], Confidence [64], τ association [65], Inf. dependency [68]	Exact match	Relational	Error/Outlier admitting [19], Query answering [64], Query rewriting [66], [67], Horizontal decomposition [69]	DD
PuD [21]	Impurity			Approximate classification [21]	-
PD [24]	Probability [24], Compression [25]			DB compression [26]	-
NuD [22]	Domain cardinality			Schema design [22]	IP
soft FD [27]	Probability			Query optimization [27], [28], [70]–[72], Table compression [73]	DD
pFD [29]	Probability			Pay-as-you-go data integration [29]	VP, DD
CD [30]	Constraint	Exact match	Relational	Query optimization [30]	IP
CFD [74]	Pattern tableau	Exact match	Relational	Data cleaning [74]–[77], [78]–[83], Data quality [32], [84]–[86], [87]–[91], Error detection [92], Materialization of data cube [93], Conflict resolution [94]	All
eCFD [11]				Data cleaning [11], [79], [82]–[84], Data quality [32], [86], [87], [90]	IP, CC, DV
CFD^p [36]				Data quality [36]	IP, CC, DV
CFD^c [59]		Synonym relation	Relational	Data quality [59]	IP, CC
MFD [38]	-	Metric distance	Relational	Source merging [38], Data quality [84]	VP
PAC [58]	Probability	Differential function	Relational	Data quality in network DBs [58]	-
ND [39]	-	Closeness function	Relational	Data mining [39]	DD
FFD [5], [15]	-	Similarity function	Fuzzy	Imprecise Data [5], [15], [95], Data quality [84]	IP, CC, DD
SFD [41]	-	Tolerance relation	Relational	Behaviour analysis [41]	-
TMFD [8]	-	Similarity function	Relational	Multimedia DB normalization [8]	-
MD [12]	-	Matching operator and similarity function	Relational	Record matching [12], [32], [44], Data cleaning [43], [80]–[82], Data quality [86], [87], [90], [91], Linked dataspaces [96], Entity resolution [83]	IP, DV, DD
CMD [62]	Constraint			Record linkage [62]	DV, DD
DD [46]	-	Differential function	Relational	Query optimization, data partition, record linkage [46]	All
ADD [60]	Confidence or g_3 error			Query optimization, data partition, record linkage [60]	IP, VP, DD
CoD [45]	-	Matching op., similarity and metric functions	Dataspace	Consistent query answering, object identication [45]	DV, VP
ACoD [45]	Confidence or g_3 error				DV, VP, DD
OD [47]	-	Order relation	Relational	Index space reduction, query optimization [47], Data quality [86], [90]	IP, CC
OD$_K$ [48]	-			Index space reduction, query optimization [48]	-
OD$_{AE}$ [48]	Disparity			Index space reduction, query optimization [48]	-
OFD [49], [50]	-	Order relation	Ordered	Semantic orderings [50]	IP
preFD [51]	-			Consistency of preference semantics [51]	
POD [52], [53]	-		Relational	Query optimization [52], [53]	
SD [54]	-	Order relation and distance function	Ordered	Knowledge discovery [54], Data quality [86], [90]	-
CSD [54]	Pattern tableau			Knowledge discovery [54], Data quality [89], [90]	-
TD [55], [56]	-	Temporal constraint	Temporal	Knowledge discovery [55], [56], Temporal trends [97]	IP, CC, DD
RUD [57]	-	Order relation	Relational	Data cube reduction, DB design [57]	DD
XCFD [37]	Constraint	Exact match	XML	XML data consistency [37]	DD
$\sigma\theta$XFD [63]	Percentage	Similarity function		XML semantic correlation [63]	DD

*IP = implication problem, CC = consistency checking, DV = detecting violation, VP = validation problem, DD = discovery from data

useful issues, such as whether a dependency can be efficiently discovered from data, or if it holds on a given database instance, as detailed in the following section.

7.2 Theoretical and Computational Aspects

In what follows we describe the main theoretical and computational issues characterizing RFDs, based on which we will perform a further comparison of the surveyed RFDS. In particular, we will focus on problems like *Axiomatization*, *Consistency Checking*, *Detecting Violation*, *Validation*, and *Discovery from Data*. The first two problems refer to intrinsic characteristics and properties of the RFDS themselves, which are verifiable without a database instance, whereas the remaining ones also require a database instance. In what follows we refer to the formers as theoretical problems, and to the other ones as practical problems.

Axiomatization. As for canonical FDS, also for RFDS it is important to have the possibility of automatically deriving new dependencies from existing ones through *Inference Rules*, also known as *Axioms*. The identification of a finite, sound, and complete set of axioms, like for example the Armstrong's inference rules, is named the *Axiomatization* problem [98], which is related to the more general problem of *Implication*, i.e., deciding whether a set of dependencies logically implies a given one. This also allows us to determine whether two sets of dependencies are equivalent, or if a given set of dependencies is redundant [99]. We know that for the canonical FD the implication problem is decidable in linear time [3]. In what follows, we analyze the surveyed RFDS and their classes based on the Axiomability property, highlighting the differences with respect to Armstrong's rules defined for the canonical FD.

In general, most of the surveyed RFDS introduce new sets of inference rules, either by extending Armstrong's rules, or by defining completely new ones when a simple extension would not guarantee the completeness property. Moreover, although some RFDS do not explicitly provide inference rules, in most cases it has been sufficient using Armstrong's rules (e.g., AFDS), or those provided by an RFD with similar characteristics. As an example, the inference rules defined for DDS [46] can also be used for their hybrid extension ADDS, as demonstrated in [60].

For RFDS relaxing on the *extent* by means of a *coverage measure* it is possible to apply Armstrong's rules by simply proving the bound of satisfiability of the used measure, on each rule from the Armstrong's set, as done for the g_3 error coverage measure [20]. In particular, for such measure it has been proved that the error of an RFD inferred through the application of an Armstrong's rule should be no larger than the ones of premise dependencies [60]. Such a proof can be used for all the RFDS using a *coverage measure* that can be reduced to the g_3 error, that is, all the RFDS expect PUDS and NUDS, whose coverage measures require the extension of the Armstrong's rules.

For RFDS relaxing on the *extent* by means of a *condition* the scenario is completely different. In fact, as stated by the authors of the two main RFDS in this class, namely CDS, and CFDS, a simple extension of Armstrong's axioms is not sufficient to guarantee the completeness property [30], [74]. Thus, for them it has been necessary to define new inference rules, and prove that they were *sound* and *complete*. However, it has also been proven that the *Implication Problem* for these RFDS has NP time complexity, hence it is solvable in polynomial time by a non-deterministic machine, but in special cases it reduces to deterministic polynomial one [30], [74]. Moreover, the set of axioms defined for CFDS also apply to RFDS derived from them, namely eCFDS [11] and CFD^{p}s [36], even though the implication analysis for them is $co-NP\ complete$, as shown by the authors [11], [36]. Thus, a non-deterministic machine can only determine what RFD cannot be implied, in polynomial time. Similar arguments apply to the hybrid extension CFD^c [59].

Concerning RFDS relaxing on the *attribute comparison* by means of an *approximate match*, only in few cases it has been possible to merely extend Armstrong's rules, like in the case of FFDS and TMFDS [5], [8]. In most cases, it has been necessary to define new inference rules to account for the greater complexity induced by the method used to quantify the similarity or the diversity of attribute values, like for MDS [12], DDS [46], and CODS [45] (even though in the last case it has not been proven the *completeness* of the defined set of axioms). Moreover, for hybrid extensions of these RFDS, those relaxing on the *extent* through a *coverage measure* simply use the set of axioms of the RFD they extend [45], [60], whereas in the case of CMDS, which relax on the *extent* through a *condition*, it has been necessary to redefine the inference rules in order to capture the specific semantics of the dependency [62]. The authors of this RFD explicitly state that the new inference rules might not be complete, referring such investigation to future works. For most of the RFDS relaxing on the *attribute comparison* by means of an *ordering criteria*, new inference rules have been defined to capture the greater expressiveness power, like in the case of ODS [47], OFDS [50], PODS [52], preFDS [51], and TDS [55]. For the remaining RFDS surveyed in this paper, to the best of our knowledge, the *Axiomatization* problem has not been addressed.

Consistency checking. As far as the *Consistency Checking* problem is concerned, it consists in the determining whether a set of axioms is consistent. In other words, checking the consistency means determining whether there can exist a non-empty database instance satisfying the given set of FDS [74]. However, in the context of RFDS the consistency checking problem is not a recurrent one, since for some of them it is not possible to define dependencies that are inconsistent.

Among the surveyed RFDS, eight of them explicitly address the consistency checking problem. In particular, for DDS, TDS, CFDS, CFD^{c}s, eCFDS, and CFD^{p}s the problem has turned out to be $NP-complete$, even though for the first four of them it reduces to polynomial complexity in special cases. For the FFDS the complexity is constant, whereas for ODS it is linear.

Detecting violation. Another property that is interesting to verify for an RFD is the *Detecting Violation*, which consists in verifying whether a database instance violates a given set of RFDS. This problem has been investigated for the following seven RFDS: CFDS, eCFDS, MDS, CMDS, DDS, CODS, and ACODS. For CFDS and eCFDS there exist exact algorithms solving the problem in polynomial time, whereas for the remaining five, all relaxing on the *attribute comparison*, the problem is $NP-complete$. However, for CMDS and DDS there exist approximate algorithms with polynomial complexity [46], [62].

Validation problem. This problem consists in verifying whether an RFD is satisfied on a database instance, and has been investigated for the following seven RFDS: pFDS, CFDS, MFDS, DDS, ADDS, CODS, and ACODS. We can observe that for the two RFDS relaxing only on the *extent*, namely pFDS and CFDS, a solution can be found in polynomial time. Similar considerations apply to DDS which can be validated in quadratic time. For the remaining ones the problem is

$NP-complete$, even though approximate algorithms have been proposed for MFDs, ADDs, and ACODs, in order to achieve a quadratic complexity.

Discovery from data. A critical property for an RFD to be used in practice is the possibility to discover it from data. In fact, although the canonical FD was introduced for database design activities, many efforts have been devoted to their discovery from data, which is an $NP-complete$ problem. Thus, the availability of efficient and/or approximate algorithms capable of reducing the execution time represents a valuable information for choosing the right RFD. Approximate algorithms solving this problem in polynomial time have been proposed for NDs, FFDs, MDs [100], TDs, and RUDs. A survey on discovery methods for RFDs is provided in [13].

From the results described above, we can observe that older RFDs (e.g., CDs, NUDs, FFDs, ODs, OFDs) are usually analyzed more from a theoretical point of view, but they often lack solutions for solving practical problems. On the contrary, more recent RFDs (e.g., AFDs, softFDs, pFDs, CFDs, NDs, CODs, RUDs) have been introduced for solving specific problems, for which it would also be necessary to have algorithms for discovering them from data. Thus, most of them are equipped with discovery algorithms, and only few of them tackle theoretical issues. Among these, it is worth to mention CFDs, among the RFDs relaxing on the *extent*, and DDs, among those relaxing on the *attribute comparison*, since they address both theoretical and practical aspects.

8 Conclusions and Future Work

In this work we have analyzed the different types of RFDs in a systematic way, highlighting commonalities and differences. In addition, we also carried out a detailed analysis of how the several types of RFDs tackle main issues and problems related to data dependency.

Our analysis has also outlined the main issues that should be addressed in future research in this area. In particular, it is necessary to apply RFDs to more application contexts, in order to improve the quality of data, or to check compliance of processes to existing standards.

In the future, further research is needed on algorithms for discovering RFDs from data, or to improve their performances, which entails investigating advanced pruning methods, so that uninteresting dependencies can be discarded. Finally, concerning RFDs based on approximate matching functions, it will be necessary to investigate sound and complete inference rules taking into account the different kind of operators, and the possible thresholds. Moreover, the application of RFDs to different application domains requires the capability of inferring suitable approximate comparison functions from examples, since general purpose functions might not work properly.

References

[1] E. F. Codd, "A relational model of data for large shared data banks," *Commun. ACM*, vol. 13, no. 6, pp. 377–387, 1970.

[2] E. F. Codd, *Further Normalization of the Data Base Relational Model*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1972, pp. 33–64.

[3] R. Fagin, "Multivalued dependencies and a new normal form for relational databases," *ACM Trans. Database Syst.*, vol. 2, no. 3, pp. 262–278, 1977.

[4] J. Rissanen, "Theory of relations for databases—A tutorial survey," in *Proc. 39th Int. Symp. Math. Found. Comput. Sci.*, 1978, pp. 536–551.

[5] K. V. S. V. N. Raju and A. K. Majumdar, "Fuzzy functional dependencies and lossless join decomposition of fuzzy relational database systems," *ACM Trans. Database Syst.*, vol. 13, no. 2, pp. 129–166, 1988.

[6] M. Arenas and L. Libkin, "A normal form for XML documents," *ACM Trans. Database Syst.*, vol. 29, no. 1, pp. 195–232, 2004.

[7] M.-L. Lee, T. W. Ling, and W. L. Low, "Designing functional dependencies for XML," in *Proc. 8th Int. Conf. Extending Database Technol.*, 2002, pp. 124–141.

[8] S.-K. Chang, V. Deufemia, G. Polese, and M. Vacca, "A normalization framework for multimedia databases," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 12, pp. 1666–1679, Dec. 2007.

[9] V. Vianu, "Dynamic functional dependencies and database aging," *J. ACM*, vol. 34, no. 1, pp. 28–59, 1987.

[10] U. Nambiar and S. Kambhampati, "Mining approximate functional dependencies and concept similarities to answer imprecise queries," in *Proc. 7th Int. Workshop Web Databases*, 2004, pp. 73–78.

[11] P. Bohannon, W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis, "Conditional functional dependencies for data cleaning," in *Proc. Int. Conf. Data Eng.*, 2007, pp. 746–755.

[12] W. Fan, H. Gao, X. Jia, J. Li, and S. Ma, "Dynamic constraints for record matching," *VLDB J.*, vol. 20, pp. 495–520, 2011.

[13] J. Liu, J. Li, C. Liu, and Y. Chen, "Discover dependencies from data—A review," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 2, pp. 251–264, Feb. 2012.

[14] M. A. Casanova, R. Fagin, and C. H. Papadimitriou, "Inclusion dependencies and their interaction with functional dependencies," in *Proc. ACM Symp. Principles Database Syst.*, 1982, pp. 171–176.

[15] M. İ. Sözat and A. Yazici, "A complete axiomatization for fuzzy functional and multivalued dependencies in fuzzy database relations," *Fuzzy Sets Syst.*, vol. 117, no. 2, pp. 161–181, 2001.

[16] Y. Huhtala, J. Kärkkäinen, P. Porkka, and H. Toivonen, "Efficient discovery of functional and approximate dependencies using partitions," in *Proc. Int. Conf. Data Eng.*, 1998, pp. 392–401.

[17] M. W. Vincent, J. Liu, and C. Liu, "Strong functional dependencies and their application to normal forms in XML," *ACM Trans. Database Syst.*, vol. 29, no. 3, pp. 445–462, 2004.

[18] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios, "Duplicate record detection: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 1, pp. 1–16, Jan. 2007.

[19] Y. Huhtala, J. Kärkkäinen, P. Porkka, and H. Toivonen, "TANE: An efficient algorithm for discovering functional and approximate dependencies," *Comput. J.*, vol. 42, no. 2, pp. 100–111, 1999.

[20] J. Kivinen and H. Mannila, "Approximate inference of functional dependencies from relations," *Theor. Comput. Sci.*, vol. 149, no. 1, pp. 129–149, 1995.

[21] D. A. Simovici, D. Cristofor, and L. Cristofor, "Impurity measures in databases," *Acta Informatica*, vol. 38, no. 5, pp. 307–324, 2002.

[22] J. Grant and J. Minker, "Normalization and axiomatization for numerical dependencies," *Inf. Control*, vol. 65, no. 1, pp. 1–17, 1985.

[23] P. C. Kanellakis, "On the computational complexity of cardinality constraints in relational databases," *Inf. Process. Lett.*, vol. 11, no. 2, pp. 98–101, 1980.

[24] S. Russell, *The Use of Knowledge in Analogy and Induction*, ser. Research notes in artificial intelligence. New York, NY, USA: Pitman, 1989.

[25] B. Pfahringer and S. Kramer, "Compression-based evaluation of partial determinations," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 1995, pp. 234–239.

[26] S. Kramer and B. Pfahringer, "Efficient search for strong partial determinations," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 1996, pp. 371–374.

[27] I. F. Ilyas, V. Markl, P. Haas, P. Brown, and A. Aboulnaga, "CORDS: Automatic discovery of correlations and soft functional dependencies," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2004, pp. 647–658.

[28] H. Kimura, G. Huo, A. Rasin, S. Madden, and S. B. Zdonik, "Correlation maps: A compressed access method for exploiting soft functional dependencies," *Proc. VLDB Endowment*, vol. 2, pp. 1222–1233, 2009.

[29] D. Z. Wang, X. L. Dong, A. D. Sarma, M. J. Franklin, and A. Y. Halevy, "Functional dependency generation and applications in pay-as-you-go data integration systems," in *Proc. 7th Int. Workshop Web Databases*, 2009, pp. 1–6.

[30] M. J. Maher, "Constrained dependencies," *Theor. Comput. Sci.*, vol. 173, no. 1, pp. 113–149, 1997.

[31] G. Cong, W. Fan, F. Geerts, X. Jia, and S. Ma, "Improving data quality: Consistency and accuracy," in *Proc. Int. Conf. Very Large Databases*, 2007, pp. 315–326.

[32] W. Fan, "Dependencies revisited for improving data quality," in *Proc. ACM Symp. Principles Database Syst.*, 2008, pp. 159–170.

[33] G. Cormode, L. Golab, K. Flip, A. McGregor, D. Srivastava, and X. Zhang, "Estimating the confidence of conditional functional dependencies," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2009, pp. 469–482.

[34] L. Bravo, W. Fan, F. Geerts, and S. Ma, "Increasing the expressivity of conditional functional dependencies without extra complexity," in *Proc. Int. Conf. Data Eng.*, 2008, pp. 516–525.

[35] L. Golab, H. Karloff, F. Korn, D. Srivastava, and B. Yu, "On generating near-optimal tableaux for conditional functional dependencies," *Proc. VLDB Endowment*, vol. 1, no. 1, pp. 376–390, 2008.

[36] W. Chen, W. Fan, and S. Ma, "Analyses and validation of conditional dependencies with built-in predicates," in *Proc. 20th Int. Conf. Database Expert Syst. Appl.*, 2009, vol. 5690, pp. 576–591.

[37] L. T. H. Vo, J. Cao, and W. Rahayu, "Discovering conditional functional dependencies in XML data," in *Proc. Australasian Database Conf.*, 2011, pp. 143–152.

[38] N. Koudas, A. Saha, D. Srivastava, and S. Venkatasubramanian, "Metric functional dependencies," in *Proc. Int. Conf. Data Eng.*, 2009, pp. 1275–1278.

[39] R. Bassée and J. Wijsen, "Neighborhood dependencies for prediction," in *Proc. Adv. Knowl. Discovery Data Min.*, 2001, pp. 562–567.

[40] G. Chen, "Fuzzy functional dependencies and a series of design issues of fuzzy relational databases," in *Fuzziness in Database Management Systems*. Heidelberg, Germany: Physica Verlag, 1995, pp. 166–185.

[41] J. Baixeries, M. Kaytoue, and A. Napoli, "Computing similarity dependencies with pattern structures," in *Proc. Int. Conf. Concept Lattices Their Appl.*, 2013, pp. 33–44.

[42] J. Baixeries, "Computing functional dependencies with pattern structures," in *Proc. Int. Conf. Concept Lattices Appl.*, 2012, pp. 175–186.

[43] L. Bertossi, S. Kolahi, and L. Lakshmanan, "Data cleaning and query answering with matching dependencies and matching functions," *Theory Comput. Syst.*, vol. 52, no. 3, pp. 441–482, 2013.

[44] K. Amshakala and N. R., "Using fuzzy logic for product matching," in *Proc. Comput. Intell., Cyber Security Comput. Models*, 2014, vol. 246, pp. 171–179.

[45] S. Song, L. Chen, and P. S. Yu, "Comparable dependencies over heterogeneous data," *VLDB J.*, vol. 22, no. 2, pp. 253–274, 2013.

[46] S. Song and L. Chen, "Differential dependencies: Reasoning and discovery," *ACM Trans. Database Syst.*, vol. 36, p. 16, 2011.

[47] S. Ginsburg and R. Hull, "Order dependency in the relational model," *Theor. Comput. Sci.*, vol. 26, nos. 1/2, pp. 149–195, 1983.

[48] J. Dong and R. Hull, "Applying approximate order dependency to reduce indexing space," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 1982, pp. 119–127.

[49] W. Ng, "Ordered functional dependencies in relational databases," *Inf. Syst.*, vol. 24, no. 7, pp. 535–554, 1999.

[50] W. Ng, "An extension of the relational data model to incorporate ordered domains," *ACM Trans. Database Syst.*, vol. 26, no. 3, pp. 344–383, 2001.

[51] W. Ng, "Preference functional dependencies for managing choices," in *Proc. 25th Int. Conf. Conceptual Model.*, 2006, pp. 140–154.

[52] J. Szlichta, P. Godfrey, and J. Gryz, "Fundamentals of order dependencies," *Proc. VLDB Endowment*, vol. 5, no. 11, pp. 1220–1231, 2012.

[53] J. Szlichta, P. Godfrey, J. Gryz, and C. Zuzarte, "Expressiveness and complexity of order dependencies," *Proc. VLDB Endowment*, vol. 6, no. 14, pp. 1858–1869, Sep. 2013.

[54] L. Golab, H. Karloff, F. Korn, A. Saha, and D. Srivastava, "Sequential dependencies," *Proc. VLDB Endowment*, vol. 2, no. 1, pp. 574–585, 2009.

[55] J. Wijsen, "Reasoning about qualitative trends in databases," *Inf. Syst.*, vol. 23, no. 7, pp. 463–487, 1998.

[56] J. Wijsen, "Trends in databases: Reasoning and mining," *IEEE Trans. Knowl. Data Eng.*, vol. 13, no. 3, pp. 426–438, May/Jun. 2001.

[57] T. Calders, R. T. Ng, and J. Wijsen, "Searching for dependencies at multiple abstraction levels," *ACM Trans. Database Syst.*, vol. 27, no. 3, pp. 229–260, 2002.

[58] F. Korn, S. Muthukrishnan, and Y. Zhu, "Checks and balances: Monitoring data quality problems in network traffic databases," in *Proc. Int. Conf. Very Large Databases*, 2003, pp. 536–547.

[59] W. Chen, W. Fan, and S. Ma, "Incorporating cardinality constraints and synonym rules into conditional functional dependencies," *Inf. Process. Lett.*, vol. 109, no. 14, pp. 783–789, 2009.

[60] S. Song, "Data dependencies in the presence of difference," Ph.D. dissertation, The Hong Kong Univ., Hong Kong, 2010.

[61] G. Cormode, L. Golab, K. Flip, A. McGregor, D. Srivastava, and X. Zhang, "Estimating the confidence of conditional functional dependencies," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2009, pp. 469–482.

[62] S. Song, L. Chen, and J. X. Yu, "Extending matching rules with conditions," in *Proc. 8th Int. Workshop Quality Databases*, 2010, pp. 13–17.

[63] F. Fassetti and B. Fazzinga, "Approximate functional dependencies for XML data," in *Proc. 11th East Eur. Conf. Adv. Databases Inf. Syst.*, 2007, pp. 86–95.

[64] R. Gummadi, A. Khulbe, A. Kalavagattu, S. Salvi, and S. Kambhampati, "SMARTINT: Using mined attribute dependencies to integrate fragmented web databases," *J. Intell. Inf. Syst.*, vol. 38, no. 3, pp. 575–599, 2012.

[65] C. Giannella and E. Robertson, "On approximation measures for functional dependencies," *Inf. Syst.*, vol. 29, no. 6, pp. 483–507, 2004.

[66] G. Wolf, A. Kalavagattu, H. Khatri, R. Balakrishnan, B. Chokshi, J. Fan, Y. Chen, and S. Kambhampati, "Query processing over incomplete autonomous databases: Query rewriting using learned data dependencies," *VLDB J.*, vol. 18, no. 5, pp. 1167–1190, 2009.

[67] G. Wolf, H. Khatri, Y. Chen, and S. Kambhampati, "QUIC: A system for handling imprecision & incompleteness in autonomous databases (demo)," in *Proc. 3rd Biennial Conf. Innovative Data Syst.*, 2007, pp. 263–268.

[68] M. M. Dalkilic and E. L. Robertson, "Information dependencies," in *Proc. ACM Symp. Principles Database Syst.*, 2000, pp. 245–253.

[69] P. D. Bra and J. Paredaens, "An algorithm for horizontal decompositions," *Inf. Process. Lett.*, vol. 17, no. 2, pp. 91–95, 1983.

[70] P. J. Haas, I. F. Ilyas, G. M. Lohman, and V. Markl, "Discovering and exploiting statistical properties for query optimization in relational databases: A survey," *Statist. Anal. Data Mining*, vol. 1, no. 4, pp. 223–250, 2009.

[71] R. Krauthgamer, A. Mehta, V. Raman, and A. Rudra, "Greedy list intersection," in *Proc. Int. Conf. Data Eng.*, 2008, pp. 1033–1042.

[72] S. Ewen, M. Ortega-Binderberger, and V. Markl, "A learning optimizer for a federated database management system," *Inf.—Forsch. und Entwick.*, vol. 20, no. 3, pp. 138–151, 2005.

[73] M. Paradies, C. Lemke, H. Plattner, W. Lehner, K.-U. Sattler, A. Zeier, and J. Kruger, "How to juggle columns: An entropy-based approach for table compression," in *Proc. 14th Int. Database Eng. Appl. Symp.*, 2010, pp. 205–215.

[74] W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis, "Conditional functional dependencies for capturing data inconsistencies," *ACM Trans. Database Syst.*, vol. 33, no. 2, p. 6, 2008.

[75] W. Fan, F. Geerts, and X. Jia, "A revival of integrity constraints for data cleaning," *Proc. VLDB Endowment*, vol. 1, no. 2, pp. 1522–1523, 2008.

[76] W. Fan, "Semandaq: A data quality system based on conditional functional dependencies," *Proc. VLDB Endowment*, vol. 1, no. 2, pp. 1460–1463, 2008.

[77] M. Yakout, A. K. Elmagarmid, J. Neville, and M. Ouzzani, "GDR: A system for guided data repair," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2010, pp. 1223–1226.

[78] W. Fan, J. Li, S. Ma, N. Tang, and W. Yu, "Interaction between record matching and data repairing," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2011, pp. 469–480.

[79] Z. Bia and M. Shan, "Review of data dependencies in data repair," *J. Inf. Comput. Sci.*, vol. 9, no. 15, pp. 4623–4630, 2012.

[80] X. Chu, I. F. Ilyas, and P. Papotti, "Holistic data cleaning: Putting violations into context," in *Proc. Int. Conf. Data Eng.*, 2013, pp. 458–469.

[81] F. Geerts, G. Mecca, P. Papotti, and D. Santoro, "The LLUNATIC data-cleaning framework," *Proc. VLDB Endowment*, vol. 6, no. 9, pp. 625–636, 2013.
[82] M. Dallachiesa, A. Ebaid, A. Eldawy, A. K. Elmagarmid, I. F. Ilyas, M. Ouzzani, and N. Tang, "NADEEF: A commodity data cleaning system," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2013, pp. 541–552.
[83] L. Bertossi and L. Bravo, "Generic and declarative approaches to data cleaning: Some recent developments," in *Handbook of Data Quality-Research and Practice*. New York, NY, USA: Springer-Verlag, 2013.
[84] L. Jiang, A. Borgida, T. Topaloglou, and J. Mylopoulos, "Data quality by design: A goal-oriented approach," in *Proc. 12th Int. Conf. Info Quality*, 2007, pp. 249–263.
[85] F. Chiang and R. J. Miller, "Discovering data quality rules," *Proc. VLDB Endowment*, vol. 1, no. 1, pp. 1166–1177, 2008.
[86] W. Fan and F. Geerts, "Uniform dependency language for improving data quality," *IEEE Data Eng. Bull.*, vol. 34, no. 3, pp. 34–42, Sep. 2011.
[87] W. Fan, F. Geerts, and X. Jia, "Conditional dependencies: A principled approach to improving data quality," in *Proc. 26th Brit. Nat. Conf. Databases*, 2009, pp. 8–20.
[88] L. Golab, T. Johnson, J. S. Seidel, and V. Shkapenyuk, "Stream warehousing with DataDepot," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2009, pp. 847–854.
[89] L. Golab, H. J. Karloff, F. Korn, and D. Srivastava, "Data auditor: Exploring data quality and semantics using pattern tableaux," *Proc. VLDB Endowment*, vol. 3, no. 2, pp. 1641–1644, 2010.
[90] L. Golab, F. Korn, and D. Srivastava, "Efficient and effective analysis of data quality using pattern tableaux," *IEEE Data Eng. Bull.*, vol. 34, no. 3, pp. 26–33, Sep. 2011.
[91] W. Fan, F. Geerts, S. Ma, N. Tang, and W. Yu, "Data quality problems beyond consistency and deduplication," in *In Search of Elegance in the Theory and Practice of Computation: Essays Dedicated to Peter Buneman*. New York, NY, USA: Springer, 2013, pp. 237–249.
[92] N. Tang, J. Li, W. Fan, and W. Yu, "Incremental detection of inconsistencies in distributed data," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 6, pp. 1367–1383, Jun. 2014.
[93] E. Garnaud, S. Maabout, and M. Mosbah, "Functional dependencies are helpful for partial materialization of data cubes," *Ann. Math. Artif. Intell.*, vol. 73, pp. 245–274, 2013.
[94] W. Fan, F. Geerts, N. Tang, and W. Yu, "Inferring data currency and consistency for conflict resolution," in *Proc. Int. Conf. Data Eng.*, 2013, pp. 470–481.
[95] P. Bosc, D. Dubois, and H. Prade, "Fuzzy functional dependencies and redundancy elimination," *J. Amer. Assoc. Inf. Sci.*, vol. 49, no. 3, pp. 217–235, 1998.
[96] U. Hassan, S. O'Riain, and E. Curry, "Leveraging matching dependencies for guided user feedback in linked data applications," in *Proc. 9th Int. Workshop Inf. Integr. Web*, 2012, pp. 1–6.
[97] C. Combi and R. Rossato, "Representing trends and trend dependencies with multiple granularities," in *Proc. 13th Int. Symp. Temporal Representation Reasoning*, 2006, pp. 3–10.
[98] L. Bravo, W. Fan, and S. Ma, "Extending dependencies with conditions," in *Proc. Int. Conf. Very Large Databases*, 2007, pp. 243–254.
[99] C. Beeri and M. Y. Vardi, "A proof procedure for data dependencies," *J. ACM*, vol. 31, no. 4, pp. 718–741, 1984.
[100] S. Song and L. Chen, "Efficient discovery of similarity constraints for matching dependencies," *Data Knowl. Eng.*, vol. 87, pp. 146–166, 2013.

Loredana Caruccio received the BSc (cum laude) and MSc (cum laude) degrees in computer science from the University of Salerno in 2009 and 2012, respectively. She is currently working toward the PhD degree in management & information technology at the University of Salerno. Her research interests include data science and web engineering.

Vincenzo Deufemia received the laurea degree (cum laude) and the PhD degree in computer science from the University of Salerno in 1999 and 2003, respectively. He was a visiting researcher at the Laboratory of Advanced enterprise Information Management Systems (AeIMS) of University of Western Sydney, Australia, in 2013. He is currently an assistant professor of computer science at the University of Salerno. His research interests include visual languages, data science, software engineering, parsing technologies, and sketch recognition. On these topics, he has published more than 100 scientific articles in referred international journals, books, and conferences. In 2011 he received a research grant from Italian Ministry of Education, University and Research (MIUR) for the project entitled "Indiana MAS and the Digital Preservation of Rock Carvings". He has served as a program committee member for many international conferences and workshops.

Giuseppe Polese received the laurea degree (cum laude) in computer science from the University of Salerno, the MSc degree in computer science from the University of Pittsburgh, and the PhD degree in applied mathematics and computer science from the University of Naples. He is an associate professor at the University of Salerno. His research interests include data science, visual languages, and web engineering. Previously, he was a project manager at the Italian Airspace Company, Alenia. He is a member of the IEEE. He has been a consultant for several software firms, including Siemens, Olivetti, and Telecom Italia.