Data Dependencies Extended for Variety and Veracity: A Family Tree

Shaoxu Song, *Member, IEEE*, Fei Gao, Ruihong Huang, and Chaokun Wang

Abstract—Besides the conventional schema-oriented tasks, data dependencies are recently revisited for data quality applications, such as violation detection, data repairing and record matching. To address the variety and veracity issues of big data, data dependencies have been extended as data quality rules to adapt to various data types, ranging from (1) categorical data with equality relationships to (2) heterogeneous data with similarity relationships, and (3) numerical data with order relationships. In this survey, we briefly review the recent proposals on data dependencies categorized into the aforesaid types of data. In addition to (a) the concepts of these data dependency notations, we investigate (b) the extension relationships between data dependencies, e.g., conditional functional dependencies (CFDs) extend the conventional functional dependencies (FDs). It forms a family tree of extensions, mostly rooted in FDs, helping us understand the expressive power of various data dependencies. Moreover, we summarize (c) the discovery of dependencies from data, since data dependencies are often unlikely to be manually specified in a traditional way, given the huge volume and high variety of big data. We further outline (d) the applications of the extended data dependencies, in particular in data quality practice. It guides users to select proper data dependencies with sufficient expressive power and reasonable discovery cost. Finally, we conclude with several directions of future studies on the emerging data.

Index Terms—Integrity constraints; data dependencies

1 Introduction

Data dependencies, such as *functional dependencies* (FDs), have been long recognized as integrity constraints in databases [24]. They are firstly utilized in database design [2]. For instance, functional dependencies (FDs) are employed to evaluate whether a relation is in third normal form (3NF) [23] or Boyce-Codd normal form (BCNF) [24]. FDs are extended to multivalued dependencies (MVDs) [6], i.e., every FD is also an MVD, in order to test whether a relation is in fourth normal form (4NF) [30]. Such data dependencies are also used in database query optimization [29], [47].

While data dependencies and their extensions are conventionally used for schema design, they have been recently revisited and extended for big data analysis. When the users are exploring data, they often experience the variety and veracity issues. To process the possibly dirty data from heterogeneous sources, data dependencies have been recently extended and used for improving data quality [31], such as error detection [102], data repairing [12], [62], data deduplication [37], etc.

Given the various data types, ranging from the conventional categorical data, numerical data to the more prevalent heterogeneous data, different extensions are made over the data dependencies with distinct expressive power. In this survey, we propose to give an entire landscape of typical data dependencies, in order to identify their relationships and distinct application scenarios. For example, if a user wants to perform data repairing over a data source with both categorical and numerical values, a direct suggestion will be DCs [98] referring to Figure 1 and Table 3.

• *KLiss, MoE; BNRist; School of Software, Tsinghua University, China. E-mail: {sxsong,gao-f16,hrh16,chaokun}@tsinghua.edu.cn*

Manuscript received XX XX, XXXX; revised XX XX, XXXX.

TABLE 1: An example relation instance r_1 of hotel

	name	address	region	star	price
t_1	New Center	No.5, Central Park	New York	3	299
t_2	New Center Hotel	No.5, Central Park	New York	3	299
t_3	St. Regis Hotel	#3, West Lake Rd.	Boston	3	319
t_4	St. Regis	#3, West Lake Rd.	Chicago, MA	3	319
t_5	West Wood Hotel	Fifth Avenue, 61st Street	Chicago	4	499
t_6	West Wood	Fifth Avenue, 61st Street	Chicago, IL	4	499
t_7	Christina Hotel	No.7, West Lake Rd.	Boston, MA	5	599
t_8	Christina	#7, West Lake Rd.	San Francisco	5	0

1.1 Background

An FD $X \rightarrow Y$, over relation R, where $X, Y \subseteq R$, states that if any two tuples in an instance of R have equal X-values, then their Y-values should also be identical.

Consider the example relation instance r_1 illustrated in Table 1. A functional dependency fd_1 below over r_1 specifies the constraint that for any two tuples of hotels, if they have the same address, then their region values must be equal,

$$\mathsf{fd}_1 : \mathsf{address} \rightarrow \mathsf{region}.$$

For instance, tuples t_1 and t_2 in Table 1, with the same address value "No.5, Central Park", have equal region value "New York" too. That is, each address is associated with precisely one region.

This fd_1 can be used to detect data quality issues in the relation instance r_1 in Table 1. For tuples t_3 and t_4 with equal values on address, they have different region values, which are then treated as a violation to the above fd_1. It implies errors occurred in t_3 or t_4, e.g., "Chicago, MA" should be "Boston" instead.

TABLE 2: An index of data dependencies with references of definition, discovery and application

Data types	Data dependencies	References of definition	References of discovery	References of application	Year	# publications using it
Categorical	SFDs, Soft Functional Dependencies	[55]	[55], [60]	[55], [60]	2004	327
	PFDs, Probabilistic Functional Dependencies	[104]	[104]	[104]	2009	55
	AFDs, Approximate Functional Dependencies	[61]	[53], [54]	[111]	1995	248
	NUDs, Numerical Dependencies	[50]		[22]	1981	18
	CFDs, Conditional Functional Dependencies	[11], [34]	[35], [36], [18], [49], [113]	[25], [40]	2007	404
	eCFDs, extended CFDs	[14]	[114]	[14]	2008	76
	MVDs, Multivalued Dependencies	[30]	[82]	[80]	1977	471
	FHDs, Full Hierarchical Dependencies	[27], [52]			1978	191
	AMVDs, Approximate MVDs	[59]			2020	1
Heterogeneous	MFDs, Metric Functional Dependencies	[64]	[64]	[64]	2009	86
	NEDs, Neighborhood Dependencies	[4]	[4]	[4]	2001	15
	DDs, Differential Dependencies	[86]	[86], [88], [89], [65]	[86], [96], [95], [93], [94]	2011	109
	CDDs, Conditional Differential Dependencies	[66]	[66]	[66]	2015	3
	CDs, Comparable Dependencies	[91], [92]	[92]	[92]	2011	18
	PACs, Probabilistic Approximate Constraints	[63]	[63]	[63]	2003	39
	FFDs, Fuzzy Functional Dependencies	[79]	[109], [108]	[13], [56], [71]	1988	496
	MDs, Matching Dependencies	[37], [33]	[85], [87], [90]	[37], [38], [41]	2009	197
	CMDs, Conditional Matching Dependencies	[110]	[110]	[110]	2017	15
Numerical	OFDs, Ordered Functional Dependencies	[76], [77]		[75]	1999	27
	ODs, Order Dependencies	[28]	[67], [99]	[28], [100]	1982	27
	DCs, Denial Constraints	[8], [9]	[19], [21], [78], [10]	[8], [9], [70], [20], [98]	2005	52
	SDs, Sequential Dependencies	[48]	[48]	[48]	2009	97
	CSDs, Conditional Sequential Dependencies	[48]	[48]	[48]	2009	97

1.2 Intuition

Owing to the variety issue of big data, real-world information often has various representation formats. As indicated in [96], the strict equality restriction limits the usage of FDs.

For example, according to fd_1, tuples t_5 and t_6 in Table 1 will be detected as a "violation", since they have "different" region values but the same address. However, "Chicago" and "Chicago, IL" indeed denote the same region in the real-world with different representation formats, i.e., no errors.

On the other hand, t_7 and t_8, which have similar addresses values but different regions, are true violations with data errors. Unfortunately, they cannot be detected by fd_1, since their address values are not exactly equal (but similar). The fd_1 considers only those tuples with the strict equality relationships on address.

Therefore, data dependency notations often need adaption to meet the requirements of various data types.

1.3 Categorization

In this article, motivated by the aforesaid veracity and variety issues of big data, we focus on recent proposals of novel data dependencies declared over various data types. Table 2 presents an overview of studied data dependencies.

(1) For the conventional **categorical data**, it is noticed that data dependencies might no longer hold over the entire set of all tuples. For example, as illustrated in Table 1, while t_1 and t_2 satisfy fd_1, t_5 and t_6 (which contain no error) do not. To support such scenarios, the important attempts are to extend data dependencies with conditions [11] or statistics [55]. The basic idea of these extensions is to make the dependencies, that originally hold for the whole table, valid only for a subset of tuples.

(2) The aforesaid extensions over categorical data still consider the equality relationship of data values. This strict constraint on equality limits the usage of data dependencies over **heterogeneous data**, since real-world information often has various representation formats or conventions, such as "Chicago" and "Chicago, IL" in Table 1. To improve the expressive power, distance metrics are introduced to data dependencies [86]. Instead of equality, data dependencies with distance/similarity metrics can specify constraints on "(dis)similar" semantics. For example, a distance constraint could state that if two tuples have similar address values, then they are similar on regions as well.

(3) Another typical data type is **numerical value**, such as star and price in Table 1. Order relationships are usually more important than the equality relationships for numerical data [28]. For instance, as illustrated in Table 1, a higher end hotel generally has a higher price.

1.4 Perspective

For each data dependency, in addition to (a) the definition of the dependency notation, we are particularly interested in other aspects, including (b) extension, how it generalizes other data dependencies; (c) discovery, how it can be obtained from data; and (d) application, how it is utilized.

1.4.1 Family Tree on Extensions

To capture the semantics over big data with variety issues, data dependencies are extended with more expressive

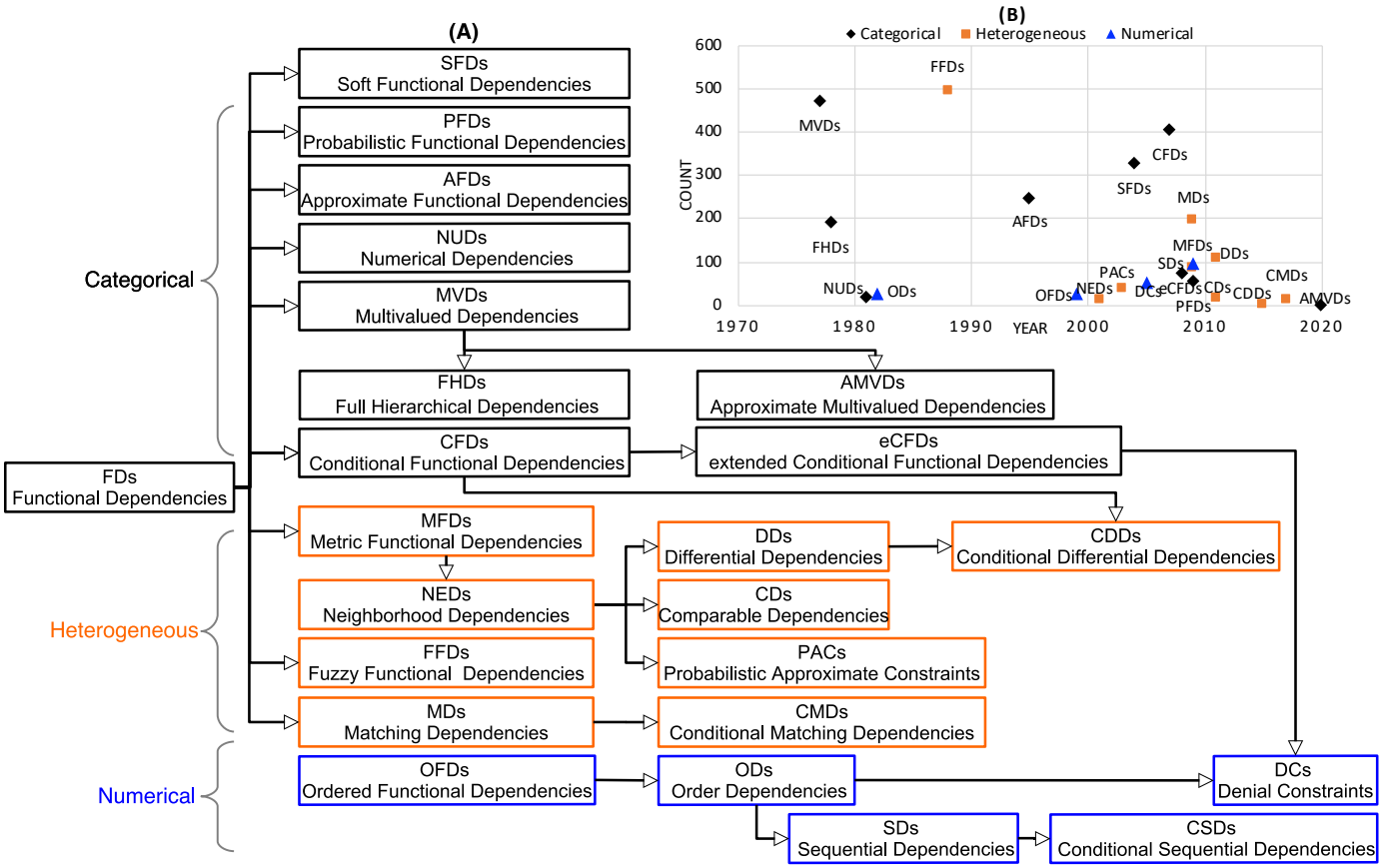


Fig. 1: (A) A family tree of extensions between various data dependencies, where an arrow (e.g., from FDs to SFDs) denotes that SFDs extend/generalize/subsume FDs, or FDs are special SFDs (with strength 1). Each extension relationship is explained in the following sections. (B) The number of publications using a data dependency

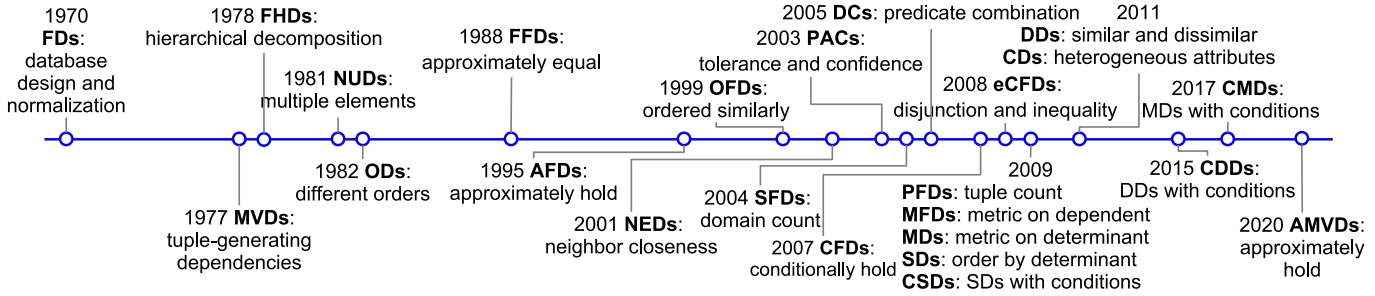


Fig. 2: Timeline of data dependencies proposed for various reasons

power. It is highly demanded to investigate the complicated extension relationships between data dependencies, in order to choose data dependencies with proper expressive power in practice. Figure 1 presents the extension relationships, where each arrow denotes the extension/generalization relationship between two types of data dependencies. For example, an arrow from FDs to SFDs denotes that SFDs extend/generalize/subsume FDs. That is, SFDs are extension of FDs, SFDs are generalization of FDs, SFDs subsume the semantics of FDs, or FDs are special SFDs (with strength 1)[1].

Table 2 and Figure 2 provide a timeline of data dependencies that are proposed for various reasons. There are some important milestones. AFDs [61] are first proposed in 1995 for data dependencies that approximately hold in a relation. Similar extensions such as SFDs [55] in 2004 and PFDs [104] in 2009 are developed following the same line. Moreover, CFDs introduce another series of extensions on conditionally holding in a relation, e.g., CSDs [11] in 2009, CDDs [66] in 2015 and CMDs [110] in 2017.

In order to compare the impact of data dependencies, Table 2 and Figure 1(B) illustrate the number of publications using a data dependency, according to Google Scholar. As shown, while the extensions over the conventional categorical data such as CFDs attract more attention, recent proposals focus more on the heterogeneous data, e.g., MDs, DDs and their extensions. Moreover, the usage of data dependencies over the numerical data is increasing, from canonical ODs to recent SDs.

1. See Section 2.1 for details.

TABLE 3: Applications of data dependencies

Applications	Categorical	Heterogeneous	Numerical
Violation detection	FDs [7], PFDs [104], CFDs [25], [40], eCFDs [14]	MFDs [64], CDs [92], CDDs [66], PACs [63]	ODs [28], DCs [8], [9], SDs [48], CSDs [48]
Data repairing	FDs [7], CFDs [25], eCFDs [14], MVDs [80]	NEDs [4], DDs [96], [95], [93], [94], CDDs [66], MDs [38], [41], CMDs[110]	DCs [70], [20], [98], ODs [28]
Query optimization	SFDs [55], [60], AFDs [111], NUDs [22], AMVDs[59]	DDs [86], CDs [92], PACs [63], FFDs [56]	ODs [100], [28]
Consistent query answering	FDs [7]	OFDs [75], DCs [8], [9]	
Data deduplication	CFDs [40]	DDs [86], CDs [92], FFDs [13], MDs [38], [41], CMDs [110]	
Data partition		DDs [86], MDs [37]	
Schema normalization	FDs [24], PFDs [104], MVDs [30], FHDs [27], [52]		
Model fairness	MVDs [80]		

1.4.2 Discovery from Data

The discovery problem is to find the data dependencies that hold in a given dataset (either clean or dirty). For instance, given the instance in Table 1, it is to find FDs such as fd_1 that (approximately) hold. The discovery of FDs from data is known to be intrinsically hard, i.e., a minimal cover can be exponential large with respect to the number of attributes in a relation [83], [72], [73]. The problem of determining whether a relation has a key of size less than a given integer is NP-complete [5]. Efficient strategies are studies for discovering FDs, such as TANE [53], [54], FastFD [112]. While the extensions with metrics enable data dependencies with more expressive power and tolerant to variations of heterogeneous data, the data dependencies become more complicated. In particular, the thresholds of metric distances/similarities are often non-trivial to specify manually in practice and thus rely more on the discovery from data. We will introduce how similar techniques are proposed for discovering various types of data dependencies.

Figure 3 compares the difficulties of data dependency discovery problems. Unfortunately, as illustrated, most problems are NP-complete. For instance, in CFD discovery, the problem of generating an optimal tableau for a given FD is NP-complete [49]. All the generalizations of CFDs, such as CDDs and DCs including CFDs as special cases in Figure 1, inherit the difficulty, i.e., the discovery of CDDs and DCs is no easier than that of CFDs. The discovery problem for CSDs, however, is polynomial time solvable, i.e., an exact dynamic programming algorithm for the tableau construction takes quadratic time in the number of candidate intervals [48].

1.4.3 Application in Data

Once the foundations of data dependencies are carefully constructed, understood and discovered, a practical issue is then how to apply such data dependencies in real data-centric applications. Besides the traditional applications, such as schema design, integrity constraints, and query optimization with respect to the schema in databases, data

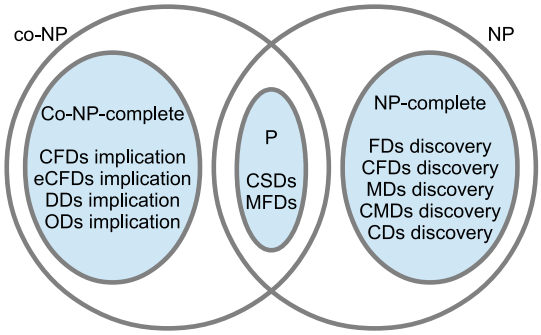


Fig. 3: The difficulties of data dependency discovery problems (assuming P≠NP and NP≠co-NP)

dependencies are recently applied in data quality-oriented practice, e.g., violation detection [11], data repairing [25], consistent query answering [3], data deduplication [37], etc. Table 3 in Section 1.4.3. The table provides the references of the data dependencies supported for each application task, again categorized by data types. It is not surprising that a large number of the studies are dedicated to the applications of violation detection and data repairing, referring to the veracity issue and the motivation of improving data quality. Moreover, the heterogeneous data type is well supported, owing to the prevalent variety issue of big data. Application details of various data dependencies are discussed in the following sections.

1.5 Difference to Existing Studies

As illustrate in Section 1.1, the tradition data dependencies such as FDs are declared on the whole relation, i.e., for any two tuples in a relation, if they have the same determinant X values, their dependent Y values must be equal as well. Such a strict condition cannot address the novel variety and veracity issues in big data. (1) Data dependencies might not hold in the whole relations. For example, one may notice that only in UK, zipcode determines street, but not in other countries. It leads to the extensions on statistics and

conditions on categorical data in Section 2. (2) The same entity may have different representation formats in various data sources. For instance, "Chicago" and "Chicago, IL" denote the same region. To be tolerant to such heterogeneity, data dependencies are extended with distance metrics over heterogeneous data in Section 3. (3) The order of numerical data needs to be considered, such as the price increases in weekdays. Extensions are necessary to capture such order constraints over numerical data in Section 4.

Caruccio et al. [15] summarize some relaxations of the traditional functional dependencies. While the relaxed notations of various data dependencies are extensively introduced, it is not discussed on how data dependencies extend with each other, i.e., the family tree of extensions presented in Figure 1. In addition to comparing FD with its more specific versions, in Figure 1(B), we further compare the impact or importance of a data dependency, by counting the number of publications that use it. To add more insights on how data dependencies relate to each other, we provide a timeline of data dependencies that are proposed for various reasons, in Figure 2. Important milestones are observed, such as AFDs [61] for data dependencies that approximately hold in a relation, and CFDs [11] on conditionally holding in a relation. Moreover, we compare the difficulties of data dependency discovery problems in Figure 3. As illustrated, while most problems are NP-complete, the discovery problem for some data dependencies such as CSDs [48], however, is polynomial time solvable.

Liu et al. [69] review typical discovery algorithms for data dependencies mainly over categorical data, such as FDs, AFDs, CFDs, etc. The discovery of data dependencies on heterogeneous and numerical data as well as their applications are not addressed.

1.6 Organization

In Section 2, we first present the data dependencies on categorical data, since conventional data dependencies are often defined on the equality relationship of categorical data values. Extensions with statistics or conditions are made to meet the variety of big data. Section 3 introduces data dependencies over heterogeneous data. The data dependencies consider similarity metrics instead of equality operators, which increase expressive power. In Section 4, data dependencies on numerical data are discussed. Orders between two values are considered as the essential constraints in these dependencies. Finally, in Section 5, we conclude the article and discuss promising directions of future work. Table 4 lists the notations frequently used in this paper.

While categorical data, heterogeneous data and numerical data are individually analyzed in each section, they would appear together in a system. Some data dependencies could express the constraints across different data types. As illustrated in Section 4.3, DCs can declare the constraints over both categorical and numerical data. For example, a DC may state that the price should not be lower than 200 (numerical value) in the region of "Chicago" (categorical value). Likewise, Section 3.3.5 shows that CDDs can express the constraints on both categorical and heterogeneous data. For instance, a CDD may state that in the region of "Chicago" (categorical value), two tuples (from heterogeneous sources) with similar name values (denoting the same hotel) should have similar address values. That is, while the data dependencies over categorical, heterogeneous and numerical data form three branches in Figure 1, they do have connections. As aforesaid, DCs extend ODs for numerical data (Section 4.3.2) as well as eCFDs on categorical data (Section 4.3.3). Similarly, CDDs extend both DDs over heterogeneous data and CFDs over categorical data (Section 3.3.5).

TABLE 4: Notations

Symbol	Description
R	relation scheme
X, Y	attribute sets in R
A, B	single attributes in R
r	relation instance
t	tuple in r
t_p	pattern tuple of conditions

2 Categorical Data

Owing to the variety issues in big data, data dependencies may not exactly hold. A natural extension is to investigate the data dependencies that almost hold, known as statistical extensions. Another idea is to extend the data dependencies with conditions, i.e., data dependencies conditionally hold in a subset of data, namely conditional extensions.

2.1 Soft Functional Dependencies (SFDs)

Unlike hard FDs, soft functional dependencies (SFDs) [55] consider that some data values determine other values not with certainty but merely with high probability, calculated by counting domain values.

2.1.1 Definition

A *soft functional dependency* (SFD) is in the form of

$$\text{SFD} : X \rightarrow_s Y$$

where X, Y are attributes in a relation R, and s is a minimum threshold of strength measure. The strength measure evaluates how $X \rightarrow Y$ almost holds in a relation instance r

$$S(X \rightarrow Y, r) = \frac{|dom(X)|_r}{|dom(X,Y)|_r},$$

where $|dom(X)|_r$ is the number of distinct values in attributes X in r, and $|dom(X,Y)|_r$ is the number of distinct values in the concatenation of X and Y in r. A SFD has $S(X \rightarrow Y, r) \geq s$, stating that the value of X determines that of Y not with certainty, but with high probability.

Consider the relation instance r_5 in Table 5. We have

$$S(\mathsf{address} \rightarrow \mathsf{region}, r_5) = \frac{|dom(X)|_{r_5}}{|dom(X,Y)|_{r_5}} = \frac{2}{3}.$$

That is, address $\rightarrow$ region almost holds in r_5. Instead, the strength measure for name $\rightarrow$ address in relation r_5 is

$$S(\mathsf{name} \rightarrow \mathsf{address}, r_5) = \frac{|dom(X)|_{r_5}}{|dom(X,Y)|_{r_5}} = \frac{1}{2}.$$

It is not a clear FD with lower strength value.

TABLE 5: An example relation instance r_5 of hotel where FD address $\rightarrow$ region almost holds, while name $\rightarrow$ address is not clear to hold

	name	address	region	rate
t_1	Hyatt	175 North Jackson Street	Jackson	230
t_2	Hyatt	175 North Jackson Street	Jackson	250
t_3	Hyatt	6030 Gateway Boulevard E	El Paso	189
t_4	Hyatt	6030 Gateway Boulevard E	El Paso, TX	189

2.1.2 Special Case: FDs

As indicated in Figure 1, SFDs extend FDs. When the value of X determines the value of Y with strength 1, it is exactly an FD. In other words, all FDs can be represented as special SFDs with $s = 1$. For example, fd_1 in Section 1.1 can be equivalently represented by

$$\mathsf{sfd}_1 : \mathsf{address} \rightarrow_1 \mathsf{region}.$$

That is, for the relation r_1 in Table 1, we have strength $S(\mathsf{address} \rightarrow \mathsf{region}, r_1) = 1$. In this sense, SFDs subsume the semantics of FDs, or SFDs generalize/extend FDs, denoted by the arrow from FDs to SFDs in Figure 1.

2.1.3 Discovery

Ilyas et al. [55] propose a sample-based approach CORDS to discover SFDs, which uses system catalog to retrieve the number of distinct values of a column. It uses a robust chi-square analysis to identify the correlation between attributes, and analyzes the number of different values in the sampling column to detect SFDs. The sample size of the algorithm is basically independent of the database size. Thereby, the algorithm is highly scalable. Kimura et al. [60] describe algorithms to search for SFDs that can be exploited at query execution time by introducing appropriate predicates and choosing a different index. It introduces bucketing on the domains of both attributes to reduce the size of index.

2.1.4 Application

Soft functional dependencies are useful in improving the optimizers selectivity estimation during query optimization [55] by collecting joint statistics for those correlated data columns. They also can be used for improving query processing performance [60] with the property that the values of an attribute are well-predicted by the values of another attribute. If a column is correlated to another column, it is possible to recommend secondary indices.

2.2 Probabilistic Functional Dependencies (PFDs)

Probabilistic functional dependencies (PFDs) [104] extend functional dependencies with probability for data integration systems, by counting the tuple values.

2.2.1 Definition

A *probabilistic functional dependency* (PFD) over attributes X and Y in relation R is denoted by

$$\text{PFD} : X \rightarrow_p Y$$

where $X \rightarrow Y$ is a standard FD, and p is a maximum threshold of the likelihood that the FD $X \rightarrow Y$ is correct.

To compute the likelihood in a relation instance r, it first calculates the fraction of tuples for each distinct value V_X of X. Let V_Y be the Y-value that occurs in the maximum number of tuples with value V_X in X. The probability is

$$P(X \rightarrow Y, V_X) = \frac{|V_Y, V_X|}{|V_X|},$$

where $|V_Y, V_X|$ is the number of tuples with values V_X for X and V_Y for Y, and $|V_X|$ is the number of tuples with values V_X for X. The probability of $X \rightarrow Y$ in r is given by the average of probabilities for each distinct value of X,

$$P(X \rightarrow Y, r) = \frac{\sum_{V_X \in D_X} P(X \rightarrow Y, V_X)}{|D_X|},$$

where D_X is all distinct values of X in r. A PFD has $P(X \rightarrow Y, r) \geq p$, i.e., a high probability to hold.

Consider again the relation instance r_5 in Table 5, where an FD address $\rightarrow$ region almost hold. We have

$$\begin{aligned} P(\mathsf{address} \rightarrow \mathsf{region}, V_1) &= 1, \\ P(\mathsf{address} \rightarrow \mathsf{region}, V_2) &= \frac{1}{2}, \\ P(\mathsf{address} \rightarrow \mathsf{region}, r_5) &= \frac{3}{4}, \end{aligned}$$

given V_1 = "175 North Jackson Street" and V_2 = "6030 Gateway Boulevard E".

Similarly, for the FD name $\rightarrow$ address, which does not clearly hold in Table 5, we have

$$P(\mathsf{name} \rightarrow \mathsf{address}, V_1) = \frac{|V_{\mathsf{address}}, V_1|}{|V_1|} = \frac{1}{2},$$

where V_1 ="Hyatt", V_{address}="6030 Gateway Boulevard E". It follows

$$P(\mathsf{name} \rightarrow \mathsf{address}, r_5) = \frac{1}{2}.$$

2.2.2 Special Case: FDs

From Figure 1, we can see that PFDs subsume FDs. When the value of X determines the value of Y with $P(X \rightarrow Y, r) = 1$, having $p = 1$ in a PFD $X \rightarrow_p Y$, it is exactly an FD. That is, all FDs can be represented as special PFDs with $p = 1$. For example, fd_1 in Section 1.1 can be represented by

$$\mathsf{pfd}_3 : \mathsf{address} \rightarrow_1 \mathsf{region}.$$

That is, for the relation r_1 in Table 1, we have $P(\mathsf{address} \rightarrow \mathsf{region}, r_1) = 1$. Consequently, PFDs subsume the semantics of FDs, or PFDs generalize/extend FDs, denoted by the arrow from FDs to PFDs in Figure 1.

2.2.3 Discovery

Wang et al. [104] extend the TANE algorithm over a single table to generate PFDs from hundreds of small, dirty and incomplete data sets. Two counting-based algorithms are proposed. The first algorithm merges the values and computes the probability of each FD, while the second algorithm, which is for multiple sources, merges PFDs obtained from each source.

2.2.4 Application

PFDs are used in a pay-as-you-go data integration system to gauge and improve the quality of the information integration [104]. FDs are declared with probabilities to capture the inherent uncertainties over many data sources. The violation of PFDs by some data sources can help pinpoint data sources with low quality data. Moreover, PFDs can also help normalize a large automatically generated mediated schema into relations that correspond to meaningful real-world entities and relationships, to help users better understand the underlying data.

2.3 Approximate Functional Dependencies (AFDs)

Rather than exactly holding, approximate functional dependencies (AFDs) [61] declares FDs that almost hold in a relation.

2.3.1 Definition

An *approximate functional dependency* (AFD) between attributes X and Y in R is denoted by

$$\text{AFD} : X \rightarrow_{\varepsilon} Y,$$

where ε is a maximum threshold of an error measure evaluating the exact proportion of tuples with violations. Given a relation instance r, the g_3 error measure calculates the ratio of the minimum number of tuples that need to be removed from r to make $X \rightarrow Y$ hold on r

$$g_3(X \rightarrow Y, r) = \frac{|r| - \max\{|s| \mid s \subseteq r, s \vDash X \rightarrow Y\}}{|r|},$$

where s is a subset of tuples in r that do not violate $X \rightarrow Y$, denoted by $s \vDash X \rightarrow Y$. An AFD has $g_3(X \rightarrow Y, r) \leq \varepsilon$. A natural interpretation is the fraction of rows with exceptions or errors affecting the dependency.

Consider again the relation instance r_5 in Table 5. We have

$$g_3(\mathsf{address} \rightarrow \mathsf{region}, r_5) = \frac{1}{4}.$$

By removing either t_3 or t_4, the violation eliminates. It can be computed by grouping tuples according to equal X values, i.e., on address, then finding the minimum violation tuples in each group.

Similarly, for name $\rightarrow$ address in r_5, we have

$$g_3(\mathsf{name} \rightarrow \mathsf{address}, r_5) = \frac{1}{2}.$$

That is, at least two tuples need to be removed (e.g., t_3 and t_4) in order to make the FD hold.

2.3.2 Special Case: FDs

Again, in Figure 1, we show that AFDs generalize FDs. For an FD, i.e., the value of X determines exactly that of Y, the corresponding error measure is $g_3(X \rightarrow Y, r) = 0$. The example fd_1 in Section 1.1 can be equivalently represented by

$$\mathsf{afd}_1 : \mathsf{address} \rightarrow_0 \mathsf{region}.$$

In other words, for the relation r_1 in Table 1, we have error measure $g_3(\mathsf{address} \rightarrow \mathsf{region}, r_1) = 0$. Thereby, AFDs subsume the semantics of FDs, or AFDs generalize/extend FDs, denoted by the arrow from FDs to AFDs in Figure 1.

2.3.3 Discovery

The TANE algorithm [53], [54] for discovering exact FDs can also be adapted for AFDs discovery. Similar to TANE, it partitions the set of rows based on attribute values for handling a large number of tuples. The use of partitions also makes the discovery of AFDs simple and efficient. The error or abnormal tuples can be easily identified. The key modification is to change the validity test on whether $X \rightarrow Y$ exactly holds to whether $g_3(X \rightarrow Y, r) \leq \varepsilon$ as defined in Section 2.3.1. It computes all minimal approximate dependencies $X \rightarrow Y$ with $g_3(X \rightarrow Y, r) \leq \varepsilon$, for a given threshold value ε.

2.3.4 Application

AFDs are utilized for query processing over incomplete databases. To retrieve possible answers, the QPIAD system [111] mines the inherent correlations among database attributes represented as AFDs. These AFDs are exploited to select features and compute probability distribution over the possible values of the missing attribute for a given tuple.

2.4 Numerical Dependencies (NUDs)

Rather than one element, numerical dependencies (NUDs) [50] state that with an element of a particular attribute or set of attributes, one can associate up to k elements of another attribute or set of attributes.

2.4.1 Definition

A *numerical dependency* (NUD) on a relation R has the form

$$\text{NUD} : X \rightarrow_k Y$$

where X, Y are attribute sets in R, and $k \geq 1$ is called the weight of the NUD. It states that each value of X can never be associated to more than k distinct values of Y.

Consider an NUD over relation r_5 in Table 5,

$$\mathsf{nud}_1 : \mathsf{address} \rightarrow_2 \mathsf{region}.$$

It states that one address can only have at most 2 variations of region. As shown in Table 5, there are 2 different region representation formats for "El Paso" in tuples t_3 and t_4.

2.4.2 Special Case: FDs

Figure 1 shows that FDs are special cases of NUDs. All FDs can be represented as special NUDs with $k = 1$. For example, fd_1 in Section 1.1 can be equivalently represented by

$$\mathsf{nud}_2 : \mathsf{address} \rightarrow_1 \mathsf{region}.$$

That is, for the relation r_1 in Table 1, each tuple $t[X]$ is associated to at most one value on Y. Thus, NUDs subsume the semantics of FDs, or NUDs generalize/extend FDs, denoted by the arrow from FDs to NUDs in Figure 1.

2.4.3 Application

NUDs can be used in various scenarios [22] such as (1) estimating the projection size of a relation, that is, number of distinct attribute values of a subset; (2) estimating the cardinality of aggregate views; and (3) efficient query processing in nondeterministic databases.

2.5 Conditional Functional Dependencies (CFDs)

Rather than FDs that hold in the whole relation, conditional functional dependencies (CFDs) [11], [34] declare FDs that conditionally hold in a part of the relation.

2.5.1 Definition

A *conditional functional dependency* (CFD) over R is a pair

$$\text{CFD} : X \rightarrow Y, t_p$$

where (1) X and Y are attributes in R; (2) $X \rightarrow Y$ is a standard FD, embedded in CFD; and (3) t_p is a pattern tuple with attributes in X and Y. For each $B \in X \cup Y$, $t_p[B]$ is either a constant 'a' in $dom(B)$, or an unnamed variable '_' that draws values from $dom(B)$. It denotes that $X \rightarrow Y$ conditionally holds over a subset of tuples specified by t_p.

A CFD over the relation instance r_5 in Table 5 can be

$$\mathsf{cfd}_1 : \mathsf{region}, \mathsf{name} \rightarrow \mathsf{address}, (\text{Jackson}, \_ \parallel \_).$$

It assures that for the tuples whose region is "Jackson", if they have the same name, then their address value must be equal (since there is only one Hyatt hotel in Jackson). For better readability, the CFD can be also written as

$$\mathsf{cfd}_1 : \mathsf{region} = \text{"Jackson"}, \mathsf{name} = \_ \rightarrow \mathsf{address} = \_.$$

Tuples t_1 and t_2 satisfy cfd_1, which have the same name, the same region of "Jackson", as well as the same address.

2.5.2 Special Case: FDs

CFDs extend FDs as shown in Figure 1. When the value of X determines the value of Y without conditions, it is exactly an FD. In other words, all FDs can be represented as special CFDs without constants in t_p. The example fd_1 in Section 1.1 can be represented as a CFD,

$$\mathsf{cfd}_2 : \mathsf{address} \rightarrow \mathsf{region}, (\_ \parallel \_).$$

For better readability, it can be also written as

$$\mathsf{cfd}_2 : \mathsf{address} = \_ \rightarrow \mathsf{region} = \_.$$

Therefore, CFDs subsume the semantics of FDs, or CFDs generalize/extend FDs, denoted by the arrow from FDs to CFDs in Figure 1.

2.5.3 Discovery

For CFDs discovery, the problem of generating an optimal tableau for a given FD is NP-complete [49]. The implication problem for CFDs is co-NP-complete [11]. Since CFDs hold only in a subset of tuples rather than the entire table as FDs, an important problem in CFD discovery is thus to evaluate how many tuples the discovered constraints can cover, known as the support of a CFD. Fan et al. [35], [36] propose three algorithms to discover CFDs. CFDMiner, based on the connection between minimal constant CFDs, finds constant CFDs by leveraging mining technique. CTANE extends TANE [53], [54] to discover general minimal CFDs, based on attribute-set/pattern tuple lattice. FastCFD, an extension of FastFD [112], discovers general CFDs by employing a depth-first search strategy. A level-wise algorithm, proposed by Chiang and Miller [18], uses an attribute lattice to generate candidate embedded FDs. Golab et al. [49] propose a greedy approximation algorithm to compute a close-to-optimal tableau for a CFD when the embedded FD is given. Peter et al. [113] present Data Quality Rules Accelerator (DQRA), by gradually introducing additional conditions to some initial CFDs.

2.5.4 Application

Cong et al. [25] study the detecting and repairing methods of violations to CFDs. Two strategies are investigated to improve data consistency, or (1) directly computing a repair that satisfies a given set of CFDs, (2) incrementally finding a repair with updates to a database. Due to the hardness of repair problems, heuristic algorithms are developed as well. Fan et al. [40] investigate the propagation problem of CFDs. Given a set of CFDs on a data source, it is to determine whether or not such CFDs are still valid on the views (mappings) of the given data source. Such propagation is useful for data integration, data exchange and data cleaning. Algorithms are developed for computing the cover of CFDs that are propagated from the original source to the views.

2.5.5 Extension: Extended CFDs (eCFDs)

To substantially improve the expressive power of CFDs, conditional functional dependencies (eCFDs) [14] extend CFDs with more predicates on disjunction and inequality.

An *extended conditional functional dependency* (eCFD) is

$$\text{eCFD} : X \rightarrow Y, t_p$$

where 1) X and Y are attributes in R; 2) $X \rightarrow Y$ is an embedded standard FD; and 3) t_p is a pattern tuple with attributes in X and Y. For each A in $X \cup Y$, $t_p[A]$ is either an unnamed variable '_' that draws values from $dom(A)$, or '$\mathsf{op}\ a$', where op is one of $\{=, \neq, <, \leq, >, \geq\}$, and '$a$' is a constant in $dom(A)$. That is, more operators are employed to specify the subset of tuples where the embedded FD holds.

An eCFD over the relation instance r_5 in Table 5 can be

$$\mathsf{ecfd}_1 : \mathsf{rate}, \mathsf{name} \rightarrow \mathsf{address}, (\leq 200, \_ \parallel \_).$$

which can also be written as follows

$$\mathsf{ecfd}_1 : \mathsf{rate} \leq 200, \mathsf{name} = \text{"\_"} \rightarrow \mathsf{address} = \_.$$

It states that if two tuples such as t_3 and t_4 in Table 5 have the same rate value $\leq$ 200, then their name determines address (since small cities often have one hotel of each brand with a relatively low rate).

There are some differences in terms of extension between CFDs and eCFDs. That is, as illustrated in Figure 1, CDDs extend CFDs (in Section 3.3.5) but not eCFDs. The complexity of the implication problem for eCFDs, implying an eCFDs from a set of eCFDs, remains unchanged, i.e., coNP-complete as CFDs [14].

2.6 Multivalued Dependencies (MVDs)

While functional dependencies (FDs) rule out the existence of certain tuples, known as equality-generating dependencies, multivalued dependencies (MVDs) [30] require the presence of certain tuples, i.e., tuple-generating dependencies.

2.6.1 Definition

A *multivalued dependency* (MVD) over relation R is in the form of

$$\text{MVD} : X \twoheadrightarrow Y,$$

where $X \cup Y \cup Z = R$ is a partition of R. A relation instance r of R satisfies the MVD, if a given pair of (X, Z) values has a set of Y values, which are determined only by X values and independent of Z values, i.e., $r = \pi_{XY}(r) \bowtie \pi_{XZ}(r)$.

An MVD in relation r_5 in Table 5 can be

$$\text{mvd}_1 : \text{address, rate} \twoheadrightarrow \text{region}.$$

For instance, given the (address, rate, name) values "(6030 Gateway Boulevard E, 189, Hyatt)" in t_3 and t_4, we have a set of region values {"El Paso", "El Paso, TX"} independent of name. Similarly, for t_1 with (address, rate, name) values (175 North Jackson Street, 230, Hyatt), we also have region value "Jackson" independent of name.

2.6.2 Special Case: FDs

From Figure 1, we can see that FDs are special cases of MVDs. All FDs can be represented as special MVDs. For example, fd_1 in Section 1.1 can be rewritten as

$$\text{mvd}_2 : \text{address} \twoheadrightarrow \text{region}.$$

That is, the value on address determines the set of values on region, having set size 1. Thereby, MVDs subsume the semantics of FDs, or MVDs generalize/extend FDs, denoted by the arrow from FDs to MVDs in Figure 1.

2.6.3 Discovery

Savnik et al. [82] study the discovery of MVDs from relations. It searches valid MVDs in the hypothesis space designed according to the generalization relationships between MVDs. The idea is generally similar to the level-wise search of FDs discovery [53], [54]. Two strategies for discovering MVDs are proposed. The top-down algorithm searches for the positive border of valid dependencies, from most general dependencies to more specific ones. The bottom-up algorithm firstly calculates the negative border of invalid MVDs by eliciting false dependencies.

2.6.4 Application

While MVDs are extremely important in database design, introducing fourth normal form (the original relation can be decomposed by MVDs and obtained from the new relations by taking joins) [30], Salimi et al. [80] recently introduce a novel application of MVDs to guarantee model fairness in machine learning. The training data often reflect discrimination, e.g., on race or gender, which is difficult to eliminate owing to the causal relationships among attributes. Intuitively, MVDs can be employed to capture the conditional independence. The fairness is thus reduced to a database repair problem by linking causal inference to multivalued dependencies (MVDs).

2.6.5 Full Hierarchical Dependencies (FHDs)

While multivalued dependencies (MVDs) decompose a relation into two of its projections without loss of information, the full hierarchical dependencies (FHDs) [27] further study the hierarchical decomposition of a relation into multiple relations. An FHD is an expression $X : \{Y_1, \ldots, Y_k\}$, where $X, Y_1, \ldots, Y_k \subseteq R$ form a partition of relation R. A relation instance r of R satisfies the FHD, if $r = \pi_{XY_1}(r) \bowtie \cdots \bowtie \pi_{XY_k}(r) \bowtie \pi_{X(R-XY_1 \ldots Y_k)}(r)$. That is, the FHD decomposes r into multiple new relations without loss of information. When $k = 1$, it is exactly an MVD $X \twoheadrightarrow Y_1$.

2.6.6 Approximate Multivalued Dependencies (AMVDs)

Similar to approximate functional dependencies (AFDs), the approximate multivalued dependencies (AMVDs) [59] capture MVDs that approximately or almost hold in a relation. AMVDs are defined as ε-MVDs with the accuracy threshold $\varepsilon \geq 0$. The accuracy relates to the percentage of spurious tuples that will be introduced by joining the relations decomposed referring to the MVDs. When $\varepsilon = 0$, i.e., no spurious tuple allowed, it is indeed the exact MVD.

2.7 Summary and Discussion

The extensions of SFDs [55] in Section 2.1, PFDs [104] in Section 2.2, AFDs [61] in Section 2.3, and NUDs [50] in Section 2.4 are with statistics. Instead of the FDs exactly holding in the data, it is to find almost valid FDs. That is, these FD-extensions are applicable to the workload where FDs hold in most tuples in a relation. Given more (approximate) rules, the recall of violation detection can be improved, while it may drag down the precision. Moreover, SFDs are efficient to compute by domain, while AFDs can tell a fine grain proportion of violation tuples.

Different from the aforesaid statistical extensions that are still declared over the whole relation, the extensions with conditions, i.e., CFDs [11] and eCFDs [14] in Sections 2.5, can be used in a workload that FDs hold only in a part of the relation. Unlike the approximately holding FDs, the accurately declared CFDs naturally have a high precision of violation detection. The coverage (recall), however, is limited, since the data dependencies hold only in a part of the data.

3 Heterogeneous Data

Extensions with statistics or conditions to FDs are still based on equality constraints. Despite the enhanced definitions, it is not robust enough to address the variety issues of big data. Data obtained from merging heterogeneous sources often have various representation conventions. Dataspaces [43], [51] provide a co-existing system of heterogeneous data from multiple sources. Carefully declared data dependencies over the heterogeneous data would be useful in query optimization and consistent query answering in dataspaces.

Table 6 presents some example data in different formats, where the tuples are from two heterogeneous sources s_1 and s_2. For instance, "12th St." in tuple t_5 from source s_2 and "12th Str" in tuple t_6 from source s_1 denote the same street but with different formats.

TABLE 6: An example relation instance r_6 with tuples from heterogeneous sources

	source	name	street	address	region	zip	price	tax
t_1	s_1	NC	CPark	#5, Central Park	New York	10041	299	29
t_2	s_2	NC	12th St.	#2 Ave, 12th St.	San Jose	95102	300	20
t_3	s_1	Regis	CPark	#9, Central Park	New York	10041	319	31
t_4	s_2	Chris	61st St.	#5 Ave, 61st St.	Chicago	60601	499	49
t_5	s_2	WD	12th St.	#6 Ave, 12th St.	San Jose	95102	399	27
t_6	s_1	NC	12th Str	#2 Aven, 12th St.	San Jose	95102	300	20

3.1 Metric Functional Dependencies (MFDs)

Metric functional dependencies (MFDs) [64] extend FDs with distance/similarity metrics on the dependent attributes Y when given the exactly matched (i.e., equal) values on determinant attributes X.

3.1.1 Definition

A *metric functional dependency* (MFD) over R has the form

$$\text{MFD} : X \rightarrow {}^{\delta}Y,$$

where (1) X, Y are two sets of attributes in R; (2) $\delta \geq 0$ is a threshold of metric distance on attributes in Y. The metric d is defined on the domain of Y, i.e., $d : dom(Y) \times dom(Y) \rightarrow \mathbb{R}$. A relation instance r over schema R satisfies the MFD, if any two tuples $t_1, t_2 \in r$ having $t_1[X] = t_2[X]$ must have distance $\leq \delta$ on attributes in Y.

Consider an example relation instance in Table 6. An MFD over the relation r_6 can be

$$\text{mfd}_1 : \text{name}, \text{region} \rightarrow^{500} \text{price}.$$

It states that if two tuples such as t_2 and t_6 have identical name and region, then their distance on attribute price should be ≤ 500, i.e., close rather than exactly equal.

3.1.2 Special Case: FDs

As indicated in Figure 1, MFDs extend FDs. When $\delta = 0$, an MFD states that if two tuples have equal X values, then their A values have distance 0, i.e., equal as well. It is exactly an FD. In other words, all FDs can be represented as special MFDs with $\delta = 0$. For example, fd_1 in Section 1.1 can be equivalently represented by

$$\text{mfd}_2 : \text{address} \rightarrow^{0} \text{region}.$$

That is, for any two tuples, equal address implies region value distance 0. Consequently, MFDs subsume the semantics of FDs, or MFDs generalize/extend FDs, denoted by the arrow from FDs to MFDs in Figure 1.

3.1.3 Discovery

During the MFDs discovery, a key step is to verify whether a candidate MFD holds for a given relation [64]. Similar to the computation of g_3 measure for AFDs in Section 2.3.3, it first groups all the tuples according to the LHS attributes X. For each group of tuples with equal X values, the maximum distance between any two tuples is computed, known as the diameter. Obviously, the MFD holds if the diameter has $\leq \delta$ in each group. In $O(n^2)$ time we can verify whether an MFD holds [64], where $n = |r|$ is the size of the relation r. Efficient algorithm is also studied to approximately verifying an MFD.

3.1.4 Application

MFDs are useful in violation detection. For instance, as presented in [64], one might expect a functional dependency of the form address $\rightarrow$ (latitude, longitude) to hold. It is worth noting that the variations are a natural part of the location data, and cannot be eliminated by format standardization. With MFDs, such small variations will not be detected erroneously as violations.

3.2 Neighborhood Dependencies (NEDs)

Neighborhood dependencies (NEDs) [4] declare data dependencies on the closeness of neighbor attribute values.

3.2.1 Definition

A *closeness function* $\theta_A(\cdot, \cdot)$ is associated to each attribute $A \in R$. The inputs of $\theta_A(\cdot, \cdot)$ are two values of attribute A, and the output is a number denoting the distance/similarity of the two input values.

A *neighborhood predicate* specifies thresholds of distance/similarity (the original definition is similarity, and for convenience, we use distance by default) on attributes,

$$A_1^{\alpha_1} \ldots A_n^{\alpha_n},$$

where (1) $A_i, 1 \leq i \leq n$ are attributes of a relation R; (2) $\alpha_i \geq 0, 1 \leq i \leq n$ are thresholds of distance/similarity on corresponding attributes A_i. Two tuples t_1, t_2 agree on the predicate if their similarity on each attribute A_i satisfies $\theta_{A_i}(t_1[A_i], t_2[A_i]) \geq \alpha_i$, or their distance on each attribute A_i satisfies $\theta_{A_i}(t_1[A_i], t_2[A_i]) \leq \alpha_i$.

A *neighborhood dependency* (NED) declares constraints between two neighborhood predicates

$$\text{NED} : A_1^{\alpha_1} \ldots A_n^{\alpha_n} \rightarrow B_1^{\beta_1} \ldots B_m^{\beta_m}.$$

A relation instance r satisfies the constraints, if for each pair of tuples in r that agrees on the predicate $A_1^{\alpha_1} \ldots A_n^{\alpha_n}$, they satisfy the predicate $B_1^{\beta_1} \ldots B_m^{\beta_m}$ as well.

For tuples in Table 6, a neighborhood predicate can be

$$\text{name}^1\text{address}^5.$$

It specifies distance thresholds 1 and 5 on attributes name and address, respectively. Two tuples t_2 and t_6 are said agreeing on the predicate, since their edit distances [74] have $\theta_{\text{name}}(t_2[\text{name}], t_6[\text{name}]) = 0 \leq 1$ and $\theta_{\text{address}}(t_2[\text{address}], t_6[\text{address}]) = 1 \leq 5$.

An NED with distance thresholds is declared by

$$\text{ned}_1 : \text{name}^1\text{address}^5 \rightarrow \text{street}^5.$$

It states that two tuples such as t_2 and t_6 mentioned above, having similar names and addresses, should have similar streets as well, i.e., $\theta_{\text{street}}(t_2[\text{street}], t_6[\text{street}]) = 3 \leq 5$.

3.2.2 Special Case: MFDs

In Figure 1, we can find that NEDs extend MFDs. When the distance threshold has $\alpha_i = 0$ in an NED, it denotes equality constraints on attributes X. Consider the example mfd_1 in Section 3.1.1, it can be represented by an NED as follows,

$$\mathsf{ned}_2 : \mathsf{name}^0\mathsf{region}^0 \rightarrow \mathsf{price}^{500}.$$

It states that if two tuples have the same name and region, then their distance of price should be ≤ 500. In other words, all MFDs can be represented as special NEDs with distance thresholds $\alpha_i = 0$. In this sense, NEDs subsume the semantics of FDs, or NEDs generalize/extend FDs, denoted by the arrow from FDs to NEDs in Figure 1.

3.2.3 Discovery

The NEDs discovery problem [4] is given the target right-hand-side neighborhood predicate, to find a left-hand-side neighborhood predicate such that the resulting NED has sufficient support and confidence. It is proved to be NP-hard in terms of the number of attributes.

3.2.4 Application

NEDs can be used for filling unknown values or repairing data errors, by a P-neighborhood method [4]. Let P and T denote the left-hand-side predicator attributes and the right-hand-side target attributes of an NED, respectively. The P-neighborhood method predicts the T value of a new tuple based on all existing neighbors of the tuple under the closeness on P attributes. This proposal is more intuitive, since the k-nearest-neighbor (kNN) method does not predefine an appropriate distance metric or k.

3.3 Differential Dependencies (DDs)

In addition to equality, differential dependencies (DDs) [86] capture the difference semantics, such as "similar" or "dissimilar", on both determinant and dependent attributes.

3.3.1 Definition

A *similarity/distance metric*, d_A, is associated to each attribute A, which satisfies non-negativity, identity of indiscernibles and symmetry. For example, the metric on a numerical attribute can be the absolute value of difference, i.e., $d_A(a, b) = |a - b|$. For a text attribute, we can adopt string similarity such as edit distance (see [74] for a survey).

A *differential function* $\phi[A]$ on attribute A indicates a range of metric distances, specified by $\{=, <, >, \leq, \geq\}$. Two tuples t_1, t_2 are compatible w.r.t. differential function $\phi[A]$, denoted by $(t_1, t_2) \asymp \phi[A]$, if the metric distance of t_1 and t_2 on attribute A is within the range specified by $\phi[A]$, a.k.a. satisfy/agree with the distance restriction $\phi[A]$. A differential function may also be specified on a set of attributes X, say $\phi[X]$, which denotes a pattern of differential functions (distance ranges) on all the attributes in X.

A *differential dependency* (DD) over a relation R has a form

$$\text{DD} : \phi[X] \rightarrow \phi[Y],$$

where $X, Y \subseteq R$ are attributes in R. It states that any two tuples satisfying differential function $\phi[X]$ must satisfy $\phi[Y]$.

Consider a differential dependency in Table 6,

$$\mathsf{dd}_1 : \mathsf{name}(\leq 1), \mathsf{street}(\leq 5) \rightarrow \mathsf{address}(\leq 5).$$

It states that if two tuples such as t_2 and t_6 have similar names (i.e., having edit distance 0 on name in the range of $[0, 1]$) and street values (with distance 3 in $[0, 5]$), they must share similar addresses values as well (having address value distance 1 in $[0, 5]$).

To express the semantics on "dissimilar", a differential dependency could be

$$\mathsf{dd}_2 : \mathsf{street}(\geq 10) \rightarrow \mathsf{address}(\geq 5).$$

That is, for any two tuples, e.g., t_1 and t_2, whose distance on street is 10 (≥ 10), their distance on address must be greater than 5. In other words, if the streets of two tuples are not similar, the corresponding addresses should be dissimilar.

3.3.2 Special Case: NEDs

Figure 1 indicates that DDs extend NEDs. When the differential functions in a DD express only the "similar" semantics, it is exactly an NED. The example ned_1 in Section 3.2.1 can be represented by a DD as follows,

$$\mathsf{dd}_3 : \mathsf{name}(\leq 1), \mathsf{address}(\leq 5) \rightarrow \mathsf{street}(\leq 5).$$

Thereby, DDs subsume the semantics of NEDs, or DDs generalize/extend NEDs, denoted by the arrow from NEDs to DDs in Figure 1.

3.3.3 Discovery

Song and Chen [86] first introduce the concept of minimal DDs and indicate that even the discovered minimal DDs could be exponentially large in size w.r.t. to the number of attributes. Several pruning strategies are then devised for DDs discovery. The determination of distance thresholds for differential functions is also studied in a parameter-free style [88], [89]. Moreover, Kwashie et al. [65] propose a subspace-clustering-based approach to discover DDs. The implication problem for DDs, i.e., implying a DD from a set of DDs, is co-NP-complete [86] .

3.3.4 Application

As indicated in [86], DDs can be used to rewrite queries in semantic query optimization. In data partition, DDs can reduce the number of predicates and improve the efficiency. In duplicate detection, DDs can address more matching rules by introducing various differential functions on one attribute. Furthermore, DDs are used as integrity constraints to enrich the candidates for missing data imputation [96], [95], and repairing data errors [93], [94].

3.3.5 Extension: Conditional DDs (CDDs)

Conditional different dependencies (CDDs) [66] extend DDs with conditions, to capture some potential knowledge and inconsistencies in a subset of data. While the conditions in CDDs are categorical values, the distance constraints can be declared over heterogeneous data. For instance, a CDD may state that in the region of "Chicago" (categorical value), two tuples (from heterogeneous sources) with similar name values (denoting the same hotel) should have similar address values. In this sense, CDDs extend both DDs over

heterogeneous data and CFDs over categorical data. As a generalization of CFDs, i.e., CDDs including CFDs as special cases in Figure 1, the discovery of CDDs is no easier than that of CFDs.

3.4 Comparable Dependencies (CDs)

To declare data dependencies over heterogeneous sources, comparable dependencies (CDs) [91], [92] consider the matching of both heterogeneous attribute names and values.

3.4.1 Definition

A *similarity function*

$$\theta(A_i, A_j) : [A_i \approx_{ii} A_i, A_i \approx_{ij} A_j, A_j \approx_{jj} A_j]$$

specifies a constraint on similarity of two values from attribute A_i or A_j, according to the corresponding similarity operators $\approx_{ii}, \approx_{ij}$ or $\approx_{jj}$. Here, A_i, A_j are often synonym attributes, and the similarity function comes together with the attribute matching on how their attribute values should be compared. Two tuples t_1, t_2 are said to be similar w.r.t. $\theta(A_i, A_j)$, denoted by $(t_1, t_2) \asymp \theta(A_i, A_j)$, if at least one of three similarity operators in $\theta(A_i, A_j)$ evaluates to true.

A *comparable dependency* (CD) is in the form of

$$\text{CD} : \bigwedge \theta(A_i, A_j) \rightarrow \theta(B_i, B_j)$$

where $\theta(A_i, A_j)$ and $\theta(B_i, B_j)$ are similarity functions. It states that for any two tuples t_1, t_2 that are similar w.r.t. $\theta(A_i, A_j)$, it implies $(t_1, t_2) \asymp \theta(B_i, B_j)$ as well.

For instance, consider a dataspace with 3 tuples,

t_1 :{name : Alice, region : Petersburg, addr : #7 T Avenue};
t_2 :{name : Alice, city : St Petersburg, post : #7 T Avenue};
t_3 :{name : Alex, region : St Petersburg, post : No 7 T Ave}.

A similarity function specified on two attributes region and city can be

$$\theta(\text{region, city}) : [\text{region} \approx_{\leq 5} \text{region}, \text{region} \approx_{\leq 5} \text{city}, \text{city} \approx_{\leq 5} \text{city}].$$

Two tuples agree the similarity function if either their region values have distance ≤ 5, or their city values have distance ≤ 5, or the region of a tuple is similar to the city of the other with distance ≤ 5. For instance, t_1 and t_2 have region and city value distance $2 \leq 5$, and thus agree θ(region, city).

Likewise, a similarity function on addr and post can be

$$\theta(\text{addr, post}) : [\text{addr} \approx_{\leq 7} \text{addr}, \text{addr} \approx_{\leq 9} \text{post}, \text{post} \approx_{\leq 5} \text{post}].$$

Tuples t_2 and t_3 having post values with distance $5 \leq 5$ again satisfy θ(region, city).

Consequently, a CD can be declared as

$$\text{cd}_1 : \theta(\text{region, city}) \rightarrow \theta(\text{addr, post}).$$

It states that if the region or city values of two tuples are similar, e.g., t_1 and t_2, then their corresponding addr or post values should also be similar.

3.4.2 Special Case: NEDs

Figure 1 shows that NEDs in Section 3.2 are special cases of CDs. When the similarity functions in a CD are defined on attributes in one table, it is exactly an NED. For instance, we can represent the example ned_1 in Section 3.2.1 by a CD,

$$\text{cd}_2 : \theta(\text{name}), \theta(\text{address}) \rightarrow \theta(\text{street}),$$

where

$$\theta(\text{name}) : [\text{name} \approx_{\leq 1} \text{name}],$$
$$\theta(\text{address}) : [\text{address} \approx_{\leq 5} \text{address}],$$
$$\theta(\text{street}) : [\text{street} \approx_{\leq 5} \text{street}].$$

It states that if the values of name and address are similar, then their corresponding street values should be similar as well. Therefore, CDs subsume the semantics of NEDs, or CDs generalize/extend NEDs, denoted by the arrow from NEDs to CDs in Figure 1.

3.4.3 Discovery

Song et al. [92] introduce a pay-as-you-go approach for discovering comparable dependencies in a given dataspace. The algorithm is conducted in an incremental way with respect to new identified comparison functions, i.e. given a set of currently discovered dependencies and a newly identified comparison functions $\theta(A_i, A_j)$, it generates new dependencies w.r.t. $\theta(A_i, A_j)$. For CDs discovery, both the error validation problem to determine whether $g_3 \leq e$, and the confidence validation problem to determine whether $conf \geq c$, are NP-complete [91], where g_3 measures the minimum number of tuples that have to be removed for the dependency to hold, and $conf$ evaluates the maximum number of tuples such that the dependency holds.

3.4.4 Application

Various applications of CDs are studied in [92]. The most important application is to improve the dataspace query efficiency. According to the comparison functions, the query evaluation searches not only the given attributes in a query tuple, e.g., region, but also their comparable attributes such as city. According to the comparable dependency, if LHS attributes of the query tuple and a data tuple are found comparable, then the data tuple can be returned without evaluating on RHS attributes. It thus improves the query efficiency. In addition, CDs can be also used to improve data quality, such as detecting violation on heterogeneous data, and identifying duplicate tuples from various data sources.

3.5 Probabilistic Approximate Constraints (PACs)

Probabilistic approximate constraints (PACs) [63] bring together distance metrics and probability. It introduces tolerance and confidence factors into integrity constraints.

3.5.1 Definition

A *probabilistic approximate constraints* (PAC)

$$\text{PAC} : X_\Delta \rightarrow^\delta Y_\epsilon$$

specifics that, if two tuples have distances on attributes X

$$|t_i[A_l] - t_j[A_l]| \leq \Delta_l, \forall A_l \in X,$$

then their probability of distances on attributes Y should be

$$\Pr(|t_i[B_l] - t_j[B_l]| \leq \epsilon_l) \geq \delta, \forall B_l \in Y,$$

where $|t_i[A_l] - t_j[A_l]|$ denotes the distance between tuples t_i and t_j on attribute A_l, $|t_i[B_l] - t_j[B_l]|$ is the distance on attribute B_l, Δ_l and ϵ_l are distance tolerances on attributes A_l and B_l, respectively, and δ is a confidence requirement.

Consider again the example in Table 6. The PAC with $\delta = 0.9$ below can tolerate that there are 10% of tuples not satisfying the distance constraints

$$\mathsf{pac}_1 : \mathsf{price}_{100} \rightarrow^{0.9} \mathsf{tax}_{10}.$$

Table 6 doesn't satisfy this PAC. There are 11 tuple pairs which have price distances less than or equal to 100. Among them, 3 pairs of tuples have tax distances greater than 10. It follows $\Pr(|t_i[\mathsf{tax}] - t_j[\mathsf{tax}]| \leq 10) = 8/11 = 0.727 < \delta$.

3.5.2 Special Case: NEDs

PACs extend NEDs in Section 3.2 as presented in Figure 1. When the probability threshold in a PAC is $\delta = 1$, it is exactly an NED. In other words, all NEDs can be represented as special PACs with $\delta = 1$. The example ned_1 in Section 3.2.1 can be expressed as a PAC as follows,

$$\mathsf{pac}_2 : \mathsf{name}_1\mathsf{address}_5 \rightarrow^1 \mathsf{street}_5.$$

It states that when two tuples have similar names and addresses, and the streets should also be similar. And $\delta = 1$ means that we don't tolerate any tuples that do not satisfy the constraints. Consequently, PACs subsume the semantics of NEDs, or PACs generalize/extend NEDs, denoted by the arrow from NEDs to PACs in Figure 1.

3.5.3 Discovery

The PAC Manager, namely PAC-Man in [63], provides a method to specify PACs. There are several parameters that need to be determined in a PAC, including Δ, ϵ for approximation with distances, and δ for the probability of satisfying the constraint. Given a set of rule-templates provided by users and some training data, PAC-Man first instantiates the aforesaid parameters. Moreover, it keeps on monitoring the new data overtime and alarm when violations are detected.

3.5.4 Application

The aforesaid PAC-Man is integrated into a database for various applications [63]. It works as a browser or monitor that keeps on tracking data quality. When problems are detected, e.g., with missing values, instead of data cleaning, PAC-Man proposes to automatically rewrite users' queries over the remaining complete observations.

3.6 Fuzzy Functional Dependencies (FFDs)

Fuzzy functional dependencies (FFDs) [79] extend FDs by replacing the equality comparison on domain values with "approximately equal", "more or less equal", etc.

3.6.1 Definition

For the domain of each attribute A_i, $dom(A_i)$, a fuzzy resemblance relation EQUAL $\mu_{EQ}(a, b), a, b \in dom(A_i)$, is defined to compare the elements of the domain, e.g., within the range of $[0, 1]$. It should be appropriately selected during database creation to capture the meaning of equality, or approximate equality, of domain values. For instance, the more the values a and b are "equal", the larger the $\mu_{EQ}(a, b)$ is (see example below). The fuzzy relation EQUAL is then extended over all attributes in R of tuples t_1 and t_2,

$$\mu_{EQ}(t_1, t_2) = \min\{\mu_{EQ}(t_1[A_1], t_2[A_1]), \mu_{EQ}(t_1[A_2], t_2[A_2]), \ldots, \mu_{EQ}(t_1[A_n], t_2[A_n])\}.$$

A *fuzzy functional dependency* (FFD),

$$\text{FFD} : X \rightsquigarrow Y$$

with $X, Y \subseteq R$, holds in a fuzzy relation instance r on R, if for all tuples t_1 and t_2 of r, we have

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

Here, $\mu_{EQ}(t_1[X], t_2[X])$ is the fuzzy resemblance relation EQUAL of tuples t_1 and t_2 on attributes X. And $\leq$ means that the resemblance relation EQUAL of tuples t_1 and t_2 on attributes X is less than that of Y. It denotes that the values on attributes Y are more "equal" than those on attributes X.

For the relation r_6 in Table 6, we consider an FFD

$$\mathsf{ffd}_1 : \mathsf{name}, \mathsf{price} \rightsquigarrow \mathsf{tax},$$

where EQUAL is defined as follows

$$\mu_{EQ}(a, b) = \begin{cases} 0 & \text{if } a \neq b \\ 1 & \text{if } a = b \end{cases}, a, b \in dom(\mathsf{name});$$
$$\mu_{EQ}(a, b) = 1/(1 + \beta|a - b|), \text{where}$$
$$\beta = \begin{cases} 1 & \text{if } a, b \in dom(\mathsf{price}) \\ 10 & \text{if } a, b \in dom(\mathsf{tax}). \end{cases}$$

It states that for any two tuples having "equal" price should have "equal" tax.

Consider two tuples t_1 and t_2 in Table 6. The EQUAL function is computed by

$$\mu_{EQ}(\mathsf{NC}, \mathsf{NC}) = 1, \quad \mathsf{NC} \in dom(\mathsf{name});$$
$$\mu_{EQ}(299, 300) = \frac{1}{1 + |299 - 300|} = \frac{1}{2}, 299, 300 \in dom(\mathsf{price});$$
$$\mu_{EQ}(29, 20) = \frac{1}{1 + 10 \times |29 - 20|} = \frac{1}{91}, 29, 20 \in dom(\mathsf{tax}).$$

It follows

$$\min(\mu_{EQ}(\mathsf{NC}, \mathsf{NC}), \mu_{EQ}(299, 300)) > \mu_{EQ}(29, 20).$$

That is, they conflict the fuzzy functional dependency, since

$$\min(\mu_{EQ}(t_1[\mathsf{name}], t_2[\mathsf{name}]), \mu_{EQ}(t_1[\mathsf{price}], t_2[\mathsf{price}])) > \mu_{EQ}(t_1[\mathsf{tax}], t_2[\mathsf{tax}]).$$

3.6.2 Special Case: FDs

As shown in Figure 1, FFDs extend FDs. For the example fd_1 in Section 1.1, by defining EQUAL as

$$\mu_{EQ}(a,b) = \begin{cases} 0 & \text{if } a \neq b \\ 1 & \text{if } a = b \end{cases}, a, b \in dom(\mathsf{address}),$$

$$\mu_{EQ}(a,b) = \begin{cases} 0 & \text{if } a \neq b \\ 1 & \text{if } a = b \end{cases}, a, b \in dom(\mathsf{region}),$$

we can obtain a corresponding FFD,

$$\mathsf{ffd}_2 : \mathsf{address} \rightsquigarrow \mathsf{region}.$$

It states that if two tuples t_1 and t_2 have the same address, i.e., $\mu_{EQ}(t_1[\mathsf{address}], t_2[\mathsf{address}]) = 1$, they must have the same region, i.e., $\mu_{EQ}(t_1[\mathsf{address}], t_2[\mathsf{address}]) = 1 \leq \mu_{EQ}(t_1[\mathsf{region}], t_2[\mathsf{region}])$. Therefore, FFDs subsume the semantics of FDs, or FFDs generalize/extend FDs, denoted by the arrow from FDs to FFDs in Figure 1.

3.6.3 Discovery

Wang et al. [109] propose a mining algorithm for FFDs, which is an extension of the TANE algorithm [53], [54]. Similar to the small-to-large search strategy in TANE (only for non-trivial FDs), it finds each non-trivial FFD with a single attribute in its right-hand-side. The algorithm checks every tuple pair to see if it satisfies the EQUAL relation. Moreover, an incremental searching algorithm is further devised based on pair-wise comparison [108]. When a new tuple is added, it avoids re-scanning of the database.

3.6.4 Application

Analogous to FDs to identify the occurrence of redundancy in a database, FFDs are also used for redundancy elimination [13], with tolerance to slight difference in the data. Intan et al. [56] studies query processing in a fuzzy relation in the presence of FFDs. Approximate join over multiple fuzzy relations is also introduced. Ma et al. [71] investigate the strategies and approaches for compressing fuzzy values by FFDs. The idea is to eliminate the unnecessary elements from a fuzzy value and thus compress its range.

3.7 Matching Dependencies (MDs)

As a matching rule, matching dependencies (MDs) [37], [33] consider similarity metrics on determinant attributes X to determine the identification of dependent attributes Y.

3.7.1 Definition

A *matching dependency* (MD) on a relation R has a form[2]

$$\text{MD} : X \approx\!\!\rightarrow Y \rightleftharpoons,$$

where (1) $X \subseteq R, Y \subseteq R$; (2) $\approx$ denotes the corresponding *similarity operator* on attributes of X, which indicates that two values are similar; (3) $\rightleftharpoons$ denotes the *matching operator* on attributes of Y, which shows that two values are identified. It states that for any two tuples from an instance of relation R, if they are similar on attributes X, then their Y values should be identified.

Consider the relation r_6 in Table 6, An MD can be

$$\mathsf{md}_1 : \mathsf{street} \approx, \mathsf{region} \approx\!\!\rightarrow \mathsf{zip} \rightleftharpoons.$$

It states that if any two tuples, e.g., t_5 and t_6 from r_6 in Table 6, have similar streets (with edit distance ≤ 5 denoted by $\approx$) and similar regions (with distance ≤ 2 denoted by $\approx$), then they can be identified (denoted by $\rightleftharpoons$) on zip.

3.7.2 Special Case: FDs

Figure 1 states that MDs subsume FDs in Section 1.1. When the values have matching similarity equal to 1.0, it is exactly an FD. The example fd_1 in Section 1.1 can thus be represented by a special MD

$$\mathsf{md}_2 : \mathsf{address} =\!\!\rightarrow \mathsf{region} =,$$

when address and region have identical values, respectively. In this sense, MDs subsume the semantics of FDs, or MDs generalize/extend FDs, denoted by the arrow from FDs to MDs in Figure 1.

3.7.3 Discovery

Song et al. [85], [87] propose both exact and approximation algorithms for discovering MDs. The exact algorithm traverses all the data to find MDs that satisfy the required confidence and support, i.e., evaluations of MDs. The approximation algorithm only traverses the first k tuples in statistical distribution, with bounded relative errors on support and confidence of returned MDs. Moreover, similar to minimal/candidate keys about FDs, relative candidate keys (RCKs) with minimal compared attributes can remove redundant semantics. Song et al. [90] find a concise set of matching keys, which can reduce the redundancy while satisfy the coverage and validity. The problem of deciding whether there exists a set of matching keys such that $supp \geq s$, $conf \geq c$, and the size of the set is no greater than k is NP-complete [90], where $supp$ measures the proportion of distinct tuple pairs that agree on at least one of the matching keys in the set, and $conf$ is the minimum ratio of tuple pairs that satisfies a matching key in the set.

3.7.4 Application

Reasoning mechanism for deducing MDs from a set of given MDs is studied in [37]. MDs and their reasoning techniques can improve both the quality and efficiency of various record matching methods. Remarkably, record matching with MDs and data repairing with CFDs can interactively perform together [38], [41]. While matching aims to identify tuples that refer to the same real-world object, repairing is to make a database consistent by fixing data errors under integrity constraints. The interaction between record matching and data repairing can effectively help with each other.

3.7.5 Extension: Conditional MDs (CMDs)

Conditional matching dependencies (CMDs) [110] extend the matching rules of MDs with conditions, which can only be applied to a part of the relation. Analogous to CFDs extending FDs in Section 2.5, CMDs binds matching dependencies only in a certain part of the table. The problem

2. Similar to CDs in Section 3.4, MDs can also be defined on attributes of two relations from heterogeneous sources [37], [33].

TABLE 7: An example relation instance r_7 with multiple numerical attributes on hotel rates

	nights	avg/night	subtotal	taxes
t_1	1	190	190	38
t_2	2	185	370	74
t_3	3	180	540	108
t_4	4	175	700	140

of deciding whether a CMD has $g_3 \leq e$ in a relation is NP-complete [110], where g_3 is the error rate of the CMD, the minimum number of tuples that need to be removed from the relation in order to make the CMD hold, and e is the maximum bound of error rate.

3.8 Summary and Discussion

To capture the relationships among heterogeneous data, instead of the equality operator in FDs over categorical data, distance and similarity of values are considered. The distance/similarity metrics could be introduced in left-hand-side, right-hand-side or both sides of attributes in a data dependency. They can also be extended to two relations from heterogeneous sources [91]. Such extensions on distance and similarity are useful not only for violation detection but also object identification [37]. Similarly, the distance metrics could also be considered over numerical data, e.g., in sequential dependencies (SDs) [48] in Section 4.4.

4 Numerical Data

Data sets with numerical values are prevalent, e.g., timestamps, sequence numbers, sales, temperature, stock prices, and so on. It is also promising to address data dependencies on such numerical data. In the following, we introduce several typical studies on such numerical data, where each attribute A has a partial ordering $\leq_A$ on $dom(A)$. Data dependencies are then declared concerning the orderings on the attribute domains [28], [44], [45], [46].

4.1 Ordered Functional Dependencies (OFDs)

Ordered functional dependency (OFD) [76], [77] state that attributes should be ordered similarly, for instance, the mileage increases with time.

4.1.1 Definition

An *ordered functional dependency* (OFD) arising from pointwise-orderings[3] over a relation R has the form

$$\text{OFD}: X \rightarrow^P Y.$$

Pointwise ordering on X, denoted by $t_1[X] \leq_X^P t_2[X]$, means that, for n attributes A_i in X, $1 \leq i \leq n, t_1[A_i] \leq t_2[A_i]$. The OFD states that for all tuples t_1, t_2 of relation R, $t_1[X] \leq_X^P t_2[X]$ implies that $t_1[Y] \leq_Y^P t_2[Y]$.

An OFD can be declared over r_7 in Table 7

$$\mathsf{ofd}_1 : \mathsf{subtotal} \rightarrow^P \mathsf{taxes}.$$

It states that a higher subtotal leads to higher taxes. For example, the subtotal of t_2 is "370", higher than that of t_1 "190". Thereby, the taxes of t_2 should be higher as well.

3. In addition to pointwise ordering, another lexicographical ordering is also introduced in [76], [77].

4.1.2 Application

OFDs are employed as useful semantic constraints over temporal relations [75]. Temporal relations can be interpreted as special cases of linearly ordered relations over time schemas. OFDs can hold the consistency of time data with various time measurement systems. For example, the experience of an employee should be increased with the passage of time.

4.2 Order Dependencies (ODs)

Order dependencies (ODs) [28] are introduced to express the constraints with different orders. For instance, the price of a production drops with the increase of time.

4.2.1 Definition

For each attribute A, the marked attributes of A are used to denote various orderings, such as $A^<$, $A^\leq$, $A^>$, $A^\geq$, and so on. For any two tuples t_1, t_2, $t_1[A^\leq]t_2$ means $t_1[A] \leq t_2[A]$.

An *order dependency* (OD) over R is in the form of

$$\text{OD}: X \rightarrow Y,$$

where X and Y are marked attributes. We say that a relation instance r over schema R satisfies the OD, if any two tuples $t_1, t_2 \in r$, $t_1[X]t_2$ implies $t_1[Y]t_2$.

For the relation instance r_7 in Table 7, an OD can be

$$\mathsf{od}_1 : \mathsf{nights}^\geq \rightarrow \mathsf{avg/night}^\leq.$$

That is, the more nights a guest stays, the lower the average price per night (avg/night) will be. For instance, for the tuples t_1 and t_2, we have $t_1[\mathsf{nights}] = 1 \leq 2 = t_2[\mathsf{nights}]$. It leads to $t_1[\mathsf{avg/night}] = 190 \geq 185 = t_2[\mathsf{avg/night}]$.

4.2.2 Special Case: OFDs

Figure 1 indicates that ODs extend OFDs in Section 4.1. When all the marked attributes are in the form of $A^\leq$ in an OD, it is exactly an OFD. For example, ofd_1 in Section 4.1.1 can be represented as an OD,

$$\mathsf{od}_2 : \mathsf{subtotal}^\leq \rightarrow \mathsf{taxes}^\leq.$$

It also means that the taxes of a tuple should be higher than that of any other tuples with less subtotal. In this sense, ODs subsume the semantics of OFDs, or ODs generalize/extend OFDs, denoted by the arrow from OFDs to ODs in Figure 1.

4.2.3 Discovery

Langer and Naumann [67] propose an OD discovery algorithm, which traverses the lattice of permutations of attributes in a level-wise bottom-up manner. Instead of expressing ODs in a list notation, Szlichta et al. [99] express ODs with sets of attributes via a polynomial mapping into a set-based canonical form. Similar to the idea of FastFD [112], an algorithm named FASTOD is presented to discover a complete, minimal set of set-based ODs. It traverses a lattice of all possible sets of attributes in a level-wise manner, reducing the complexity from using list-based axiomatization. The implication problem for ODs, i.e., implying an OD from a set of ODs, is co-NP-complete [101].

4.2.4 Application

ODs are used for database implementation to improve efficiency [28]. Firstly, ODs can reduce indexing space without much access time increase. For instance, suppose that an employee database is sorted with rank. One can access the employee instance using the order of rank. In addition, if the database also satisfies the OD rank $\rightarrow$ salary, then the data is also ordered by salary. That is, we can get salary by rank rapidly. Moreover, ODs can also be used in query optimization. Szlichta et al. [100] present optimization techniques using ODs for various SQL functions and algebraic expressions. Finally, following the same line of other data dependencies, ODs are naturally applicable as integrity constraints for error detection and data repairing.

4.3 Denial Constraints (DCs)

In addition to the dependency between determinant and dependent attributes, Denial constraints (DCs) [8], [9] study a more general form of integrity constraints that prevent some attributes from taking certain values, by built-in atoms with $\{=, \neq, <, >, \leq, \geq\}$.

4.3.1 Definition

A *denial constraint* (DC) has a form

$$\text{DC} : \forall t_\alpha, t_\beta, \cdots \in R, \neg(P_1 \wedge \cdots \wedge P_m)$$

where P_i is of the form $v_1\ \phi\ v_2$ or $v_1\ \phi\ c$, ϕ is an element of a negation closed finite operator set $\{=, \neq, <, >, \leq, \geq\}$, and $v_1, v_2 \in t_\alpha.A, t_\beta.A, \ldots, A \in R$, and c is a constant. It states that all the predicates cannot be true at the same time.

Consider the relation instance r_7 in Table 7. The order relationships on numeric values such as subtotal and taxes can be captured by a DC with operators $<, >$

$$\begin{aligned}\text{dc}_1 : \forall t_\alpha, t_\beta \in R, \neg(t_\alpha.\text{subtotal} < t_\beta.\text{subtotal} \\ \wedge t_\alpha.\text{taxes} > t_\beta.\text{taxes}).\end{aligned}$$

It declares that a lower subtotal should not pay more taxes. For example, tuples t_1 and t_2 satisfy dc_1, having $t_1[\text{subtotal}] < t_2[\text{subtotal}]$ and $t_1[\text{taxes}] < t_2[\text{taxes}]$.

4.3.2 Special Case: ODs

Figure 1 states that DCs subsume ODs in Section 4.2. We can represent od_1 in Section 4.2.1 by a DC as follows,

$$\begin{aligned}&\text{dc}_2 : \forall t_\alpha, t_\beta \in R, \\ &\neg(t_\alpha.\text{nights} \geq t_\beta.\text{nights} \wedge t_\alpha.\text{avg/night} > t_\beta.\text{avg/night}).\end{aligned}$$

Both constraints state that any $t_\alpha, t_\beta \in R$ should *not* have $t_\alpha[\text{nights}] \geq t_\beta[\text{nights}]$ but $t_\alpha[\text{avg/night}] > t_\beta[\text{avg/night}]$, i.e., stays more nights but with a higher rate. In this sense, DCs subsume the semantics of ODs, or DCs generalize/extend ODs, denoted by the arrow from FDs to DCs in Figure 1.

4.3.3 Special Case: eCFDs

Again, Figure 1 shows that DCs extend eCFDs. We can represent the example ecfd_1 in Section 2.5.5 by a DC,

$$\begin{aligned}&\text{dc}_3 : \forall t_\alpha, t_\beta \in R, \\ &\neg(t_\alpha.\text{rate} = t_\beta.\text{rate} \wedge t_\alpha.\text{rate} \leq 200 \wedge t_\alpha.\text{name} = t_\beta.\text{name} \\ &\wedge t_\alpha.\text{address} \neq t_\beta.\text{address}).\end{aligned}$$

It states that for any $t_\alpha, t_\beta \in R$, they should *not* have same rate (≤ 200) and the same name but different addresses. Consequently, DCs subsume the semantics of eCFDs, or DCs generalize/extend eCFDs, denoted by the arrow from eCFDs to DCs in Figure 1.

4.3.4 Discovery

Chu et al. [19] present a DC discovery algorithm FASTDC as an extension of FastFD [112]. It starts by building a predicate space and calculates evidence sets. For instance, for the DCs involving at most two tuples without constants, the structure of a predicate consists of two different attributes and one operator. The algorithm establishes the connection between discovering minimal DCs and finding minimal set covers for evidence sets. Depth-first search strategy is deployed for finding minimal set covers, together with DC axioms for branch pruning. Chu et al. [19] also extend FASTDC to discover approximate DCs (A-FASTDC) and constant DCs (C-FASTDC). Since FASTDC is sensitive to the number of records in the dataset, Pena and Almeida present BFASTDC [78], a bitwise version of FASTDC that uses logical operations to form the auxiliary data structures from which DCs are mined. Bleifuß et al. [10] propose a new algorithm Hydra, which overcomes the quadratic runtime complexity in the number of tuples in a relation. Based on the FASTDC algorithm, a system for discovering DC rules is implemented, namely RuleMiner [21].

4.3.5 Application

Denial constraints are useful for detecting violations and enforcing the correct application semantics. Linear denial constraints [8], [9] are utilized to fix the numerical attributes in databases. Efficient approximation algorithms to obtain a database repair are presented [70]. Since DCs subsume existing formalisms and can express rules involving numerical values, with predicates such as "greater than" and "less than", Chu et al. [20] propose a holistic repairing algorithm under the constraints of DCs. It is worth noting that in practice, both the given DC rules and the data could be dirty. Song et al. [98] study the simultaneous repairing of both DCs and data.

4.4 Sequential Dependencies (SDs)

Sequential dependencies (SDs) [48] generalize ODs to express interesting relationships between ordered determinant attributes and distances on dependent attributes.

4.4.1 Definition

A *sequential dependency* (SD) is in the form of

$$\text{SD} : X \rightarrow_g Y,$$

where $X \subseteq R$ is a set of ordered attributes, $Y \subseteq R$ can be measured by certain distance metrics, and g is an interval. It states that when tuples are sorted on X, the distances between the Y-values of any two consecutive tuples are within interval g.

For the relation instance r_7 in Table 7, we can set an SD,

$$\text{sd}_1 : \text{nights} \rightarrow_{[100,200]} \text{subtotal}.$$

It identifies that subtotal raises within the interval $[100, 200]$ with the increase of nights. As shown in Table 7, tuples are sorted on nights. Tuples t_2 and t_3 have "2" and "3" nights, respectively. The corresponding distance/increase of subtotal is 540-370=170 within the range of $[100, 200]$.

4.4.2 Special Case: ODs

As stated in Figure 1, SDs subsume ODs in Section 4.2 as special cases. The interval g on distance in an SD can be used to express the order relationships, such as $[0, \infty)$ or $(-\infty, 0]$. Thereby, we rewrite od_1 in Section 4.2.1 as an SD,

$$\mathsf{sd}_2 : \mathsf{nights} \rightarrow_{(-\infty,0]} \mathsf{avg/night}.$$

It means that avg/night for a room decreases with the increase of nights that a guest will stay. Therefore, SDs subsume the semantics of ODs, or SDs generalize/extend ODs, denoted by the arrow from ODs to SDs in Figure 1.

4.4.3 Discovery

To discover reasonable SDs, Golab et al. [48] first define the confidence of an SD, i.e., the minimum number of tuples that needs to be removed or inserted to make the SD hold in a given dataset. It is worth noting that tuple insertion may apply to satisfy the distance constraints in an SD, different from the g_3 error measure of AFD in Section 2.3 with tuple deletion only to make an FD hold. Efficient computation of confidence is then studied for SD discovery, e.g., simple SDs of the form $X \rightarrow_{(0,\infty)} Y$ where Y always increases with X.

4.4.4 Application

SDs are useful in network monitoring, e.g., auditing the polling frequency [48]. An SD : pollnum $\rightarrow_{[9,11]}$ time could be employed. It requires that the data collector probe the counters in about every 10 seconds. Too frequent polls, with time interval less than 9, or missing data, with time interval greater than 10, may indicate problems of the collector.

4.4.5 Extension: Conditional SDs (CSDs)

Since the frequency of data feed varies with time and the measurement attributes fluctuate with time, conditional order dependence (CSD) [48] declare SDs that conditionally hold in a period. As illustrated in Figure 3 in Section 1.4.2, while most problems are NP-complete, the discovery problem for some data dependencies such as CSDs [48], however, is polynomial time solvable. That is, for CSD tableau discovery, an exact dynamic programming algorithm for the tableau construction takes quadratic time in the number of candidate intervals [48].

4.5 Summary and Discussion

For numerical data, data dependencies are extended with operators $<, \leq, >, \geq$ for expressing the order relationships between data values. The ideas of conditional constraints for categorical data and distance metrics for heterogeneous data are also employed for data dependencies over numerical data, such as CSDs with conditions on determinant attributes and distances on dependent attributes as introduced in Section 4.4.5.

5 Conclusion and Future Directions

In this survey, we briefly review a number of recent proposals on data dependencies. Most studies aim to adapt conventional data dependency notations to various types of big data. We categorize these novel data dependencies into three types, i.e., on categorical data, heterogeneous data, or numerical data. Remarkably, the extension relationships among data dependencies are investigated. It forms a family tree on extensions (mostly) rooted in functional dependencies (FDs). In addition, we also discuss how these novel data dependencies could be discovered from data. This is particularly important, since data dependencies are often unlikely to be manually specified in a traditional way, given the huge volume and high variety of big data. Moreover, we also introduce how these data dependencies could be utilized in data quality applications.

Future work may be raised over more emerging data. Novel types of data dependencies are expected to be declared on the data not stored in conventional relational databases, such as graph, time series or uncertain data.

5.1 Uncertain Data

Uncertain data are prevalent [81], e.g., owing to noises that make it deviate from the correct or original value in sensor networks [17]. The data are expected to be cleaned by reducing the uncertainty [16]. Motivated by the probability measures of functional dependencies in Section 2, a natural idea is to study the data dependencies over uncertain data. An uncertain relation allows to give multiple possible values for tuples, and represents a set of possible worlds, each of which is an ordinary relation. Sarma et al. [81] study the functional dependencies for uncertain relations. The proposed horizontal FDs and vertical FDs are consistent with the conventional FDs when an uncertain relation does not contain any uncertainty. Xiang et al. [68] study the problem of consistent query answering over uncertain data. A novel notion of repaired possible worlds (involving both repair and possible worlds) is proposed. In this sense, uncertainty could be introduced in all the aspects, naturally embedded in the data, probed in the discovered dependencies, or generated in possible repairs. It is highly demanded to holistically study these aspects of uncertainty.

5.2 Graph Data

Graph data widely exist in real-world applications such as knowledge bases, workflows or social networks [116]. Like other types of big data, veracity issues are prevalent in graph data. For example, gene ontology annotation could be erroneous in a protein interaction network [93]. The event names are misplaced in a workflow network [105]. To clean such errors, neighborhood constraints between vertices are considered, e.g., extracted from workflow specification [103]. Unlike relational data, graph data do not have a specific schema. Thereby, most of the existing dependencies cannot be applied in graph data directly. Graph data have structural characteristics that can be used for defining dependencies [39], while they also have various forms, such as XML, RDF or networks. Each form of graph may have its own features. The variety of graph data is another challenge

when solving the problems in data. Fan et al. [32], [42] present some pioneer studies in this promising direction.

5.3 Temporal Data

Temporal data are characterized by data elements with time-varying information [58]. It is important in various scenarios such as IoT, where sensor devices uninterruptedly obtain data from the physical world. The sensor devices are often unreliable with missing readings [57] or even wrong timestamps [84]. As a result, time series data are usually very large, incomplete [106], [107] and dirty [105], [115]. While data dependencies over numerical data such as sequential dependencies [48] could be employed to partially capture the order information, more advanced constraints on temporal features are expected. Speed constraints [97] are some attempts of declaring constraints on consecutive data. Unfortunately, it is not well studied yet on how to discover such meaningful speed constraints. Abedjan et al. [1] study the temporal rule discovery for web data cleaning. It is further expected to address the unique data quality challenges due to the presence of autocorrelations, trends, seasonality, and gaps in the time series data [26].

REFERENCES

[1] Z. Abedjan, C. G. Akcora, M. Ouzzani, P. Papotti, and M. Stonebraker. Temporal rules discovery for web data cleaning. *PVLDB*, 9(4):336–347, 2015.

[2] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[3] M. Arenas, L. E. Bertossi, and J. Chomicki. Consistent query answers in inconsistent databases. In *Proceedings of the Eighteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 31 - June 2, 1999, Philadelphia, Pennsylvania, USA*, pages 68–79, 1999.

[4] R. Bassée and J. Wijsen. Neighborhood dependencies for prediction. In *Knowledge Discovery and Data Mining - PAKDD 2001, 5th Pacific-Asia Conference, Hong Kong, China, April 16-18, 2001, Proceedings*, pages 562–567, 2001.

[5] C. Beeri, M. Dowd, R. Fagin, and R. Statman. On the structure of armstrong relations for functional dependencies. *J. ACM*, 31(1):30–46, 1984.

[6] C. Beeri, R. Fagin, and J. H. Howard. A complete axiomatization for functional and multivalued dependencies in database relations. In *Proceedings of the 1977 ACM SIGMOD International Conference on Management of Data, Toronto, Canada, August 3-5, 1977*, pages 47–61, 1977.

[7] L. E. Bertossi. *Database Repairing and Consistent Query Answering*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

[8] L. E. Bertossi, L. Bravo, E. Franconi, and A. Lopatenko. Complexity and approximation of fixing numerical attributes in databases under integrity constraints. In *DBPL*, volume 3774 of *Lecture Notes in Computer Science*, pages 262–278. Springer, 2005.

[9] L. E. Bertossi, L. Bravo, E. Franconi, and A. Lopatenko. The complexity and approximation of fixing numerical attributes in databases under integrity constraints. *Inf. Syst.*, 33(4-5):407–434, 2008.

[10] T. Bleifuß, S. Kruse, and F. Naumann. Efficient denial constraint discovery with hydra. *PVLDB*, 11(3):311–323, 2017.

[11] P. Bohannon, W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis. Conditional functional dependencies for data cleaning. In *Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, The Marmara Hotel, Istanbul, Turkey, April 15-20, 2007*, pages 746–755, 2007.

[12] P. Bohannon, M. Flaster, W. Fan, and R. Rastogi. A cost-based model and effective heuristic for repairing constraints by value modification. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland, USA, June 14-16, 2005*, pages 143–154, 2005.

[13] P. Bosc, D. Dubois, and H. Prade. Fuzzy functional dependencies and redundancy elimination. *JASIS*, 49(3):217–235, 1998.

[14] L. Bravo, W. Fan, F. Geerts, and S. Ma. Increasing the expressivity of conditional functional dependencies without extra complexity. In *Proceedings of the 24th International Conference on Data Engineering, ICDE 2008, April 7-12, 2008, Cancún, México*, pages 516–525, 2008.

[15] L. Caruccio, V. Deufemia, and G. Polese. Relaxed functional dependencies - A survey of approaches. *IEEE Trans. Knowl. Data Eng.*, 28(1):147–165, 2016.

[16] R. Cheng, J. Chen, and X. Xie. Cleaning uncertain data with quality guarantees. *PVLDB*, 1(1):722–735, 2008.

[17] R. Cheng and S. Prabhakar. Managing uncertainty in sensor database. *SIGMOD Record*, 32(4):41–46, 2003.

[18] F. Chiang and R. J. Miller. Discovering data quality rules. *PVLDB*, 1(1):1166–1177, 2008.

[19] X. Chu, I. F. Ilyas, and P. Papotti. Discovering denial constraints. *PVLDB*, 6(13):1498–1509, 2013.

[20] X. Chu, I. F. Ilyas, and P. Papotti. Holistic data cleaning: Putting violations into context. In *29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8-12, 2013*, pages 458–469, 2013.

[21] X. Chu, I. F. Ilyas, P. Papotti, and Y. Ye. Ruleminer: Data quality rules discovery. In *IEEE 30th International Conference on Data Engineering, Chicago, ICDE 2014, IL, USA, March 31 - April 4, 2014*, pages 1222–1225, 2014.

[22] P. Ciaccia, M. Golfarelli, and S. Rizzi. Efficient derivation of numerical dependencies. *Inf. Syst.*, 38(3):410–429, 2013.

[23] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387, 1970.

[24] E. F. Codd. Further normalization of the data base relational model. *IBM Research Report, San Jose, California*, RJ909, 1971.

[25] G. Cong, W. Fan, F. Geerts, X. Jia, and S. Ma. Improving data quality: Consistency and accuracy. In *Proceedings of the 33rd International Conference on Very Large Data Bases, University of Vienna, Austria, September 23-27, 2007*, pages 315–326, 2007.

[26] T. Dasu, R. Duan, and D. Srivastava. Data quality for temporal streams. *IEEE Data Eng. Bull.*, 39(2):78–92, 2016.

[27] C. Delobel. Normalization and hierarchical dependencies in the relational data model. *ACM Trans. Database Syst.*, 3(3):201–222, 1978.

[28] J. Dong and R. Hull. Applying approximate order dependency to reduce indexing space. In *Proceedings of the 1982 ACM SIGMOD International Conference on Management of Data, Orlando, Florida, June 2-4, 1982.*, pages 119–127, 1982.

[29] M. Eich, P. Fender, and G. Moerkotte. Faster plan generation through consideration of functional dependencies and keys. *PVLDB*, 9(10):756–767, 2016.

[30] R. Fagin. Multivalued dependencies and a new normal form for relational databases. *ACM Trans. Database Syst.*, 2(3):262–278, 1977.

[31] W. Fan. Dependencies revisited for improving data quality. In *Proceedings of the Twenty-Seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2008, June 9-11, 2008, Vancouver, BC, Canada*, pages 159–170, 2008.

[32] W. Fan, Z. Fan, C. Tian, and X. L. Dong. Keys for graphs. *PVLDB*, 8(12):1590–1601, 2015.

[33] W. Fan, H. Gao, X. Jia, J. Li, and S. Ma. Dynamic constraints for record matching. *VLDB J.*, 20(4):495–520, 2011.

[34] W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis. Conditional functional dependencies for capturing data inconsistencies. *ACM Trans. Database Syst.*, 33(2), 2008.

[35] W. Fan, F. Geerts, L. V. S. Lakshmanan, and M. Xiong. Discovering conditional functional dependencies. In *Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, March 29 2009 - April 2 2009, Shanghai, China*, pages 1231–1234, 2009.

[36] W. Fan, F. Geerts, J. Li, and M. Xiong. Discovering conditional functional dependencies. *IEEE Trans. Knowl. Data Eng.*, 23(5):683–698, 2011.

[37] W. Fan, X. Jia, J. Li, and S. Ma. Reasoning about record matching rules. *PVLDB*, 2(1):407–418, 2009.

[38] W. Fan, J. Li, S. Ma, N. Tang, and W. Yu. Interaction between record matching and data repairing. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2011, Athens, Greece, June 12-16, 2011*, pages 469–480, 2011.

[39] W. Fan and P. Lu. Dependencies for graphs. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of*

Database Systems, PODS 2017, Chicago, IL, USA, May 14-19, 2017, pages 403–416, 2017.

[40] W. Fan, S. Ma, Y. Hu, J. Liu, and Y. Wu. Propagating functional dependencies with conditions. *PVLDB*, 1(1):391–407, 2008.

[41] W. Fan, S. Ma, N. Tang, and W. Yu. Interaction between record matching and data repairing. *J. Data and Information Quality*, 4(4):16:1–16:38, 2014.

[42] W. Fan, Y. Wu, and J. Xu. Functional dependencies for graphs. In *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 1843–1857, 2016.

[43] M. J. Franklin, A. Y. Halevy, and D. Maier. From databases to dataspaces: a new abstraction for information management. *SIGMOD Record*, 34(4):27–33, 2005.

[44] S. Ginsburg and R. Hull. Order dependency in the relational model. *Theor. Comput. Sci.*, 26:149–195, 1983.

[45] S. Ginsburg and R. Hull. Sort sets in the relational model. In *Proceedings of the Second ACM SIGACT-SIGMOD Symposium on Principles of Database Systems, March 21-23, 1983, Colony Square Hotel, Atlanta, Georgia, USA*, pages 332–339, 1983.

[46] S. Ginsburg and R. Hull. Sort sets in the relational model. *J. ACM*, 33(3):465–488, 1986.

[47] T. Gogacz and S. Toruńczyk. Entropy bounds for conjunctive queries with functional dependencies. In *20th International Conference on Database Theory, ICDT 2017, March 21-24, 2017, Venice, Italy*, pages 15:1–15:17, 2017.

[48] L. Golab, H. J. Karloff, F. Korn, A. Saha, and D. Srivastava. Sequential dependencies. *PVLDB*, 2(1):574–585, 2009.

[49] L. Golab, H. J. Karloff, F. Korn, D. Srivastava, and B. Yu. On generating near-optimal tableaux for conditional functional dependencies. *PVLDB*, 1(1):376–390, 2008.

[50] J. Grant and J. Minker. Numerical dependencies. In *XP2 Workshop on Relational Database Theory, June 22-24 1981, The Pennsylvania State University, PA, USA*, 1981.

[51] A. Y. Halevy, M. J. Franklin, and D. Maier. Principles of dataspace systems. In *Proceedings of the Twenty-Fifth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 26-28, 2006, Chicago, Illinois, USA*, pages 1–9, 2006.

[52] S. Hartmann, H. Köhler, and S. Link. Full hierarchical dependencies in fixed and undetermined universes. *Ann. Math. Artif. Intell.*, 50(1-2):195–226, 2007.

[53] Y. Huhtala, J. Kärkkäinen, P. Porkka, and H. Toivonen. Efficient discovery of functional and approximate dependencies using partitions. In *Proceedings of the Fourteenth International Conference on Data Engineering, Orlando, Florida, USA, February 23-27, 1998*, pages 392–401, 1998.

[54] Y. Huhtala, J. Kärkkäinen, P. Porkka, and H. Toivonen. Tane: An efficient algorithm for discovering functional and approximate dependencies. *Comput. J.*, 42(2):100–111, 1999.

[55] I. F. Ilyas, V. Markl, P. J. Haas, P. Brown, and A. Aboulnaga. CORDS: automatic discovery of correlations and soft functional dependencies. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004*, pages 647–658, 2004.

[56] R. Intan and M. Mukaidono. Fuzzy functional dependency and its application to approximate data querying. In *2000 International Database Engineering and Applications Symposium, IDEAS 2000, September 18-20, 2000, Yokohoma, Japan, Proccedings*, pages 47–54, 2000.

[57] S. R. Jeffery, G. Alonso, M. J. Franklin, W. Hong, and J. Widom. Declarative support for sensor data cleaning. In *Pervasive Computing, 4th International Conference, PERVASIVE 2006, Dublin, Ireland, May 7-10, 2006, Proceedings*, pages 83–100, 2006.

[58] C. S. Jensen and R. T. Snodgrass. Temporal data management. *IEEE Trans. Knowl. Data Eng.*, 11(1):36–44, 1999.

[59] B. Kenig, P. Mundra, G. Prasaad, B. Salimi, and D. Suciu. Mining approximate acyclic schemes from relations. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*, pages 297–312, 2020.

[60] H. Kimura, G. Huo, A. Rasin, S. Madden, and S. B. Zdonik. Correlation maps: A compressed access method for exploiting soft functional dependencies. *PVLDB*, 2(1):1222–1233, 2009.

[61] J. Kivinen and H. Mannila. Approximate inference of functional dependencies from relations. *Theor. Comput. Sci.*, 149(1):129–149, 1995.

[62] S. Kolahi and L. V. S. Lakshmanan. On approximating optimum repairs for functional dependency violations. In *Database Theory - ICDT 2009, 12th International Conference, St. Petersburg, Russia, March 23-25, 2009, Proceedings*, pages 53–62, 2009.

[63] F. Korn, S. Muthukrishnan, and Y. Zhu. Checks and balances: Monitoring data quality problems in network traffic databases. In *VLDB 2003, Proceedings of 29th International Conference on Very Large Data Bases, September 9-12, 2003, Berlin, Germany*, pages 536–547, 2003.

[64] N. Koudas, A. Saha, D. Srivastava, and S. Venkatasubramanian. Metric functional dependencies. In *Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, March 29 2009 - April 2 2009, Shanghai, China*, pages 1275–1278, 2009.

[65] S. Kwashie, J. Liu, J. Li, and F. Ye. Mining differential dependencies: A subspace clustering approach. In *Databases Theory and Applications - 25th Australasian Database Conference, ADC 2014, Brisbane, QLD, Australia, July 14-16, 2014. Proceedings*, pages 50–61, 2014.

[66] S. Kwashie, J. Liu, J. Li, and F. Ye. Conditional differential dependencies (cdds). In *Advances in Databases and Information Systems - 19th East European Conference, ADBIS 2015, Poitiers, France, September 8-11, 2015, Proceedings*, pages 3–17, 2015.

[67] P. Langer and F. Naumann. Efficient order dependency detection. *VLDB J.*, 25(2):223–241, 2016.

[68] X. Lian, L. Chen, and S. Song. Consistent query answers in inconsistent probabilistic databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2010, Indianapolis, Indiana, USA, June 6-10, 2010*, pages 303–314, 2010.

[69] J. Liu, J. Li, C. Liu, and Y. Chen. Discover dependencies from data - a review. *IEEE Trans. Knowl. Data Eng.*, 24(2):251–264, 2012.

[70] A. Lopatenko and L. Bravo. Efficient approximation algorithms for repairing inconsistent databases. In *Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, The Marmara Hotel, Istanbul, Turkey, April 15-20, 2007*, pages 216–225, 2007.

[71] Z. M. Ma, W. Zhang, and F. Mili. Fuzzy data compression based on data dependencies. *Int. J. Intell. Syst.*, 17(4):409–426, 2002.

[72] H. Mannila and K.-J. Räihä. *Design of Relational Databases*. Addison-Wesley, 1992.

[73] H. Mannila and K.-J. Räihä. Algorithms for inferring functional dependencies from relations. *Data Knowl. Eng.*, 12(1):83–99, 1994.

[74] G. Navarro. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1):31–88, 2001.

[75] W. Ng. Lexicographically ordered functional dependencies and their application to temporal relations. In *1999 International Database Engineering and Applications Symposium, IDEAS 1999, Montreal, Canada, August 2-4, 1999, Proceedings*, pages 279–287, 1999.

[76] W. Ng. Ordered functional dependencies in relational databases. *Inf. Syst.*, 24(7):535–554, 1999.

[77] W. Ng. An extension of the relational data model to incorporate ordered domains. *ACM Trans. Database Syst.*, 26(3):344–383, 2001.

[78] E. H. M. Pena and E. C. de Almeida. BFASTDC: A bitwise algorithm for mining denial constraints. In *DEXA (1)*, volume 11029 of *Lecture Notes in Computer Science*, pages 53–68. Springer, 2018.

[79] 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.*, 13(2):129–166, 1988.

[80] B. Salimi, L. Rodriguez, B. Howe, and D. Suciu. Interventional fairness: Causal database repair for algorithmic fairness. In *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*, pages 793–810, 2019.

[81] A. D. Sarma, J. D. Ullman, and J. Widom. Schema design for uncertain databases. In *Proceedings of the 3rd Alberto Mendelzon International Workshop on Foundations of Data Management, Arequipa, Peru, May 12-15, 2009.*, 2009.

[82] I. Savnik and P. A. Flach. Discovery of multivalued dependencies from relations. *Intell. Data Anal.*, 4(3-4):195–211, 2000.

[83] J. C. Schlimmer. Efficiently inducing determinations: A complete and systematic search algorithm that uses optimal pruning. In *Machine Learning, Proceedings of the Tenth International Conference, University of Massachusetts, Amherst, MA, USA, June 27-29, 1993*, pages 284–290, 1993.

[84] S. Song, Y. Cao, and J. Wang. Cleaning timestamps with temporal constraints. *PVLDB*, 9(10):708–719, 2016.

[85] S. Song and L. Chen. Discovering matching dependencies. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009*, pages 1421–1424, 2009.

[86] S. Song and L. Chen. Differential dependencies: Reasoning and discovery. *ACM Trans. Database Syst.*, 36(3):16, 2011.

[87] S. Song and L. Chen. Efficient discovery of similarity constraints for matching dependencies. *Data Knowl. Eng.*, 87:146–166, 2013.

[88] S. Song, L. Chen, and H. Cheng. Parameter-free determination of distance thresholds for metric distance constraints. In *IEEE 28th International Conference on Data Engineering (ICDE 2012), Washington, DC, USA (Arlington, Virginia), 1-5 April, 2012*, pages 846–857, 2012.

[89] S. Song, L. Chen, and H. Cheng. Efficient determination of distance thresholds for differential dependencies. *IEEE Trans. Knowl. Data Eng.*, 26(9):2179–2192, 2014.

[90] S. Song, L. Chen, and H. Cheng. On concise set of relative candidate keys. *PVLDB*, 7(12):1179–1190, 2014.

[91] S. Song, L. Chen, and P. S. Yu. On data dependencies in dataspaces. In *Proceedings of the 27th International Conference on Data Engineering, ICDE 2011, April 11-16, 2011, Hannover, Germany*, pages 470–481, 2011.

[92] S. Song, L. Chen, and P. S. Yu. Comparable dependencies over heterogeneous data. *VLDB J.*, 22(2):253–274, 2013.

[93] S. Song, H. Cheng, J. X. Yu, and L. Chen. Repairing vertex labels under neighborhood constraints. *PVLDB*, 7(11):987–998, 2014.

[94] S. Song, B. Liu, H. Cheng, J. X. Yu, and L. Chen. Graph repairing under neighborhood constraints. *VLDB J.*, 26(5):611–635, 2017.

[95] S. Song, Y. Sun, A. Zhang, L. Chen, and J. Wang. Enriching data imputation under similarity rule constraints. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2018.

[96] S. Song, A. Zhang, L. Chen, and J. Wang. Enriching data imputation with extensive similarity neighbors. *PVLDB*, 8(11):1286–1297, 2015.

[97] S. Song, A. Zhang, J. Wang, and P. S. Yu. SCREEN: stream data cleaning under speed constraints. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, pages 827–841, 2015.

[98] S. Song, H. Zhu, and J. Wang. Constraint-variance tolerant data repairing. In *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 877–892, 2016.

[99] J. Szlichta, P. Godfrey, L. Golab, M. Kargar, and D. Srivastava. Effective and complete discovery of order dependencies via set-based axiomatization. *PVLDB*, 10(7):721–732, 2017.

[100] J. Szlichta, P. Godfrey, J. Gryz, W. Ma, W. Qiu, and C. Zuzarte. Business-intelligence queries with order dependencies in DB2. In *Proceedings of the 17th International Conference on Extending Database Technology, EDBT 2014, Athens, Greece, March 24-28, 2014.*, pages 750–761, 2014.

[101] J. Szlichta, P. Godfrey, J. Gryz, and C. Zuzarte. Expressiveness and complexity of order dependencies. *Proc. VLDB Endow.*, 6(14):1858–1869, 2013.

[102] S. Thirumuruganathan, L. Berti-Équille, M. Ouzzani, J. Quiané-Ruiz, and N. Tang. Uguide: User-guided discovery of fd-detectable errors. In *Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD Conference 2017, Chicago, IL, USA, May 14-19, 2017*, pages 1385–1397, 2017.

[103] W. M. P. van der Aalst. *Process Mining - Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011.

[104] 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 *12th International Workshop on the Web and Databases, WebDB 2009, Providence, Rhode Island, USA, June 28, 2009*, 2009.

[105] J. Wang, S. Song, X. Lin, X. Zhu, and J. Pei. Cleaning structured event logs: A graph repair approach. In *31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015*, pages 30–41, 2015.

[106] J. Wang, S. Song, X. Zhu, and X. Lin. Efficient recovery of missing events. *PVLDB*, 6(10):841–852, 2013.

[107] J. Wang, S. Song, X. Zhu, X. Lin, and J. Sun. Efficient recovery of missing events. *IEEE Trans. Knowl. Data Eng.*, 28(11):2943–2957, 2016.

[108] S. Wang, J. Shen, and T. Hong. Incremental discovery of fuzzy functional dependencies. In *Handbook of Research on Fuzzy Information Processing in Databases*, pages 615–633. 2008.

[109] X. Wang and G. Chen. Discovering fuzzy functional dependencies as semantic knowledge in large databases. In *The Fourth International Conference on Electronic Business - Shaping Business Strategy in a Networked World*, pages 1136–1139, 2004.

[110] Y. Wang, S. Song, L. Chen, J. X. Yu, and H. Cheng. Discovering conditional matching rules. *TKDD*, 11(4):46:1–46:38, 2017.

[111] G. Wolf, H. Khatri, B. Chokshi, J. Fan, Y. Chen, and S. Kambhampati. Query processing over incomplete autonomous databases. In *Proceedings of the 33rd International Conference on Very Large Data Bases, University of Vienna, Austria, September 23-27, 2007*, pages 651–662, 2007.

[112] C. M. Wyss, C. Giannella, and E. L. Robertson. Fastfds: A heuristic-driven, depth-first algorithm for mining functional dependencies from relation instances - extended abstract. In *Data Warehousing and Knowledge Discovery, Third International Conference, DaWaK 2001, Munich, Germany, September 5-7, 2001, Proceedings*, pages 101–110, 2001.

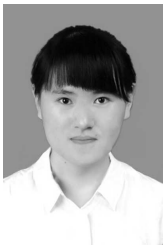
[113] P. Z. Yeh, C. A. Puri, M. Wagman, and A. K. Easo. Accelerating the discovery of data quality rules: A case study. In *Proceedings of the Twenty-Third Conference on Innovative Applications of Artificial Intelligence, August 9-11, 2011, San Francisco, California, USA*, 2011.

[114] A. Zanzi and A. Trombetta. Discovering non-constant conditional functional dependencies with built-in predicates. In *Database and Expert Systems Applications - 25th International Conference, DEXA 2014, Munich, Germany, September 1-4, 2014. Proceedings, Part I*, pages 35–49, 2014.

[115] A. Zhang, S. Song, and J. Wang. Sequential data cleaning: A statistical approach. In *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 909–924, 2016.

[116] J. Zhang, C. Wang, J. Wang, and J. X. Yu. Inferring continuous dynamic social influence and personal preference for temporal behavior prediction. *PVLDB*, 8(3):269–280, 2014.

Shaoxu Song is an associate professor in the School of Software, Tsinghua University, Beijing, China. His research interests include data quality and complex event processing. He has published more than 30 papers in top conferences and journals such as SIGMOD, VLDB, ICDE, TODS, TKDE, VLDBJ, etc.



Fei Gao is a Ph.D. candidate in the School of Software, Tsinghua University, Beijing, China. Her current research interests include time series data cleaning and data quality.

Ruihong Huang is working toward the PhD degree in the School of Software, Tsinghua University, Beijing, China. His current research interests include event data quality and time series time quality.

Chaokun Wang is an associate professor in the School of Software, Tsinghua University. He has published more than 60 refereed papers, received two best paper awards at international conferences and held 12 patents. His current research interests include social network analysis, graph data management, and music computing.