A Survey of Database Dependency Concepts

Nikita Bobrov
Saint Petersburg State University
Email: nv.mm61@gmail.com

Anastasia Birillo
Saint Petersburg State University
Email: anastasia.i.birillo@gmail.com

George Chernishev
Saint Petersburg State University,
JetBrains Research
Email: g.chernyshev@spbu.ru

***Abstract*—A database dependency is a formal concept which is used to describe patterns in data. These patterns are employed during data analysis, data cleansing, and schema normalization. There is about dozen of major dependency types.**

In this paper we survey these types of database dependencies employed in the relational databases. We start from the earliest ones — functional dependencies and conclude with the state-of-the-art findings. For each type we provide both formal and non-formal definitions and present an example and counterexample. We also briefly discuss extraction algorithms and possible use cases.

I. Introduction

The era of big data not only presents new opportunities, but also poses new challenges for both industry and academy. The challenges come from the three "Vs", which are frequently used to describe the properties of big data: volume, velocity, and variety. Volume refers to the ever-increasing amounts of data that need to be processed. Velocity usually implies rapid arrival speeds and variety reflects that different types of data involved.

To benefit from this data, one has to be able to process, store, and query it efficiently. This requires that data semantics — patterns, properties, and purposes must be known to the user. Unfortunately, most of the time this is not the case for big data: it is hard to obtain this knowledge because of these three properties, especially given the large volumes.

There is a strong demand for methods, tools, and approaches for knowledge extraction. One of such tools is database dependencies, a concept known since the early 70s. This approach is very promising because big data is closely related to the NoSQL movement, which relies on very wide "tables". Thus, knowledge extraction employing these dependencies becomes very relevant in this environment.

A database dependency is a formal concept that can be used to describe patterns in data. Initially, the dependencies were employed for schema normalization and data cleansing. Currently, one of the most popular contemporary applications is data analysis.

A database dependency can be described as a rule which has left and right hand sides (LHS and RHS). This rule guarantees some properties of the data and involves LHS and RHS. For example, functional dependency guarantees that for any pair of tuples with equal LHS their RHS would be equal also.

Database dependencies are quite an old concept. The first kind — functional dependencies were introduced in 1971. Since then, the area grew rapidly, and many novel types and sub-types were proposed. Currently, there is a dozen of major dependency types.

In this paper we survey database dependency concepts employed in the relational databases. We survey them starting from the earliest ones and conclude with the state-of-the-art findings. We provide both formal and non-formal definitions and present an example for each type. We also briefly discuss extraction algorithms and possible use cases. Our goal is to produce a broad survey that involves major types of database dependencies without delving into details.

II. Motivation

The idea of this paper arrived during our development of a data-driven tool for automatic database physical design. Our approach was to rely on data properties instead of workload knowledge. During this study an extensive exploration of dependency concept literature was performed. We have discovered many classes of such concepts, not only textbook examples like functional dependencies. However, there were no single study which would present a broad overview of this field.

Thus, we decided to convert our expertise into a survey which would cover the major classes of dependencies, including the recent ones. This survey may be useful to beginners and to researchers who are interested in discovery of knowledge hidden in the data and who do not require solid theoretical understanding.

III. Related Work

There are several surveys on database dependency concepts. However, they are not entirely comparable to this study due to a number of reasons. First, there is a couple of studies that survey different kinds of dependencies, but which are more than 20 years old [1]. This fact renders them unusable for learning up-to-date dependency types.

On the other hand, the recent studies pursue goals that differ from the ones of our survey.

Reference [2] is a classification of types of relaxed functional dependencies. In this survey, the authors consider 35 types of imprecise functional dependencies, build a classification and provide a list of application domains for each class. However, the authors deliberately left inclusion dependencies and multivalued dependencies out of scope of this work.

There is another type of surveys that deal with dependency discovery methods. Reference [3] contains a Related Work

Patient	Gender	Doctor	Hospital	Area	Phone
Benson	M	Morgan	St. Mark's	North	89084140683
Ellis	F	Robin	St. George	South	89608401913
Graham	M	Robin	St. Mark's	North	89084140683
Harris	M	Jean	St. Mungo's	West	89607968712
Joy	F	Smith	St. Thomas'	East	89685290066

TABLE I
FD EXAMPLE

section that lists a number of methods for discovery of functional dependencies. Another functional dependency discovery survey is presented in reference [4]. This work features not only a survey, but also an experimental evaluation.

A survey of discovery methods for different types of dependencies is presented in reference [5]. This survey includes methods for conditional, inclusion, approximate, and XML dependency discovery.

Thus, to the best of our knowledge, there is no broad survey which lists known types of database dependencies.

IV. Dependency Types

A. Functional Dependencies

The notion of the functional dependency (FD) was originally introduced by E.F. Codd in the early 1970s in his paper "Further Normalization of the Data Base Relational Model" [6]. In this paper, Codd proposed to use FDs for database schema design. Now, the range of FD application is much wider. For example, study [7] describes query optimization methods that are based on the FD notion.

Study [8] contains research results on the topic of FDs that are used for query optimization. Besides, the usefulness of FDs is shown in relational database systems as well as in non-relational database environments.

The first formal definition of an FD was given in W.W. Armstrong's work [9].

Definition 1: A relation R satisfies the FD $X \to Y$ (where $X, Y \subseteq R$), if and only if for all $t_1, t_2 \in R$ holds: if $t_1[X] = t_2[X]$, then $t_1[Y] = t_2[Y]$.

Thus, an FD is essentially a "many to one" relation between the value sets of attributes participating in LHS and RHS.

Consider relation depicted in Table I. The following FDs are present:

- {Area, Phone} → {Hospital}
- {Patient} → {Gender}

As we can see from Table I, any value from the set {Patient, Doctor} is related exactly to one value from the set {Area}. The second FD can be explained in a similar manner.

Further, we can list dependencies which do not hold (violate the FD condition) in the relation:

- {Doctor} → {Hospital, Area}
- {Hospital} → {Patient}

The first one is not an FD since two different values in the RHS attribute set (Hospital and Area) present for a fixed value of the Doctor attribute. For example doctor Robin maps simultaneously to $\{St.George, South\}$, $\{St.Mark's, North\}$ that contradicts the meaning of FD. The issue with the second dependency can be explained in a similar way.

Nowadays, relaxed FDs (RFDs) [2] are one of the most active sub-types of FDs. These dependencies relax one or more constraints of the canonical FD. For example, RFD has to hold for most tuples, not for all of them like in FD case.

B. Inclusion Dependencies

Inclusion dependency (IND) was first formalized by R. Fagin [10], but have also been used by J.M. Smith and D.C.P. Smith in [11]. Since then INDs have become just as popular as any other traditional dependency type (FD, MVD, JD).

By introducing this concept, a new kind of a normal form was defined — the domain-key normal form (DK/NF). This form requires every constraint on the relation to be a logical consequence of key constraints and domain constraints. A relation is in DK/NF if it is guaranteed that no insertion and deletion anomalies are present, as it is stated in Fagin's theorem 3.13 ("a satisfiable 1NF relation schema is in DK/NF if and only if it has no insertion or deletion anomalies" [10]). We say that an insertion anomaly occurs if after insertion of a tuple one of relation constraints (either key or domain) is violated. Similarly, a deletion anomaly occurs when deletion of a single tuple results in constraint violation. According to paper [12], we define IND as:

Definition 2: Let $R = (R_1 \dots R_k)$ and $S = (S_1 \dots S_m)$ be two relational tables. Further, $\hat{R} = R_{i_1} \dots R_{i_n}$ and $\hat{S} = S_{i_1} \dots S_{i_n}$ be n-ary column combinations of distinct columns. We say that IND $\hat{R} \subseteq \hat{S}$ holds, if for every tuple $t_R \in R$, there is a tuple $t_S \in S$, such that $t_R[\hat{R}] = t_S[\hat{S}]$. $\hat{R}$ is called the dependent column combination and $\hat{S}$ – the referenced column combination.

In other words, we can say that all tuples of the attribute combination $\hat{R}$ in the relation R must be also contained in tuples of the attribute combination $\hat{S}$ of the relation S.

Figure IV-B depicts an example of IND {DLN} ⊆ {DLID}. This dependency information can help us in discovery of a foreign key (which is DLN in the example).

UID	Name	Gender	DLN
1	Sofia	F	21
2	Leonard	M	35
3	Shavkat	M	10
4	Mary	F	65
5	Andrew	M	10

DLID	Country
21	Romania
35	Spain
10	Germany
65	USA

TABLE II
IND EXAMPLE

Let us construct an example where the dependency presented in Table IV-B is violated. In order to violate {DLN} ⊆ {DLID}, the DLN should stop being the foreign key for the corresponding table. Thus, we have to substitute any value of the DLN with the value absent in the $DLID$. For example, changing the third tuple to $(3, Shavkat, M, 11)$ is enough to break this dependency.

Name	Disease	Doctor
Wilson	Arthritis	Lewis
Taylor	Insomnia	Jackson
Kimberly	Pancreatitis	Brooks
Ellis	Schizophrenia	Brooks

TABLE III
JD EXAMPLE

Name	Doctor
Wilson	Lewis
Taylor	Jackson
Kimberly	Brooks
Ellis	Brooks

Name	Disease
Wilson	Arthritis
Taylor	Insomnia
Kimberly	Pancreatitis
Ellis	Somnambulism

TABLE IV
JD EXAMPLE

Name	Doctor
Wilson	Lewis
Taylor	Jackson
Kimberly	Brooks
Ellis	Brooks

Doctor	Disease
Lewis	Arthritis
Jackson	Insomnia
Brooks	Pancreatitis
Brooks	Somnambulism

TABLE V
JD COUNTEREXAMPLE

IND is one of the most important dependencies for many tasks such as data integration, integrity checking, schema design, and any other where we may need additional metadata for understanding significant aspects or structure of an unknown database. Consider a data source that comes only with superficial information, such as relation or attributes names with no interrelational constraints. In this case, detected INDs between relations may be considered as a precondition for a foreign key constraint. In fact, being a basis of the interrelational constraint is the most prominent application of the IND.

There are further state of the art approaches for exact and approximate inference of INDs: [13], [14], [12].

C. Join Dependencies

Join dependencies (JDs) were first mentioned at the end of the 1970s in the works [15], [16]. The notion of JDs ensures lossless reconstruction of the data that was decomposed into a number of relations. The data is reconstructed using the join relational operation.

Paper [17] contains the formal definition of JD:

Definition 3: A JD defined for a relation $R[U]$ (U — is a set of attributes) is an expression of the form $\bowtie [X_1, \cdots, X_n]$, where $X_1 \cup \cdots \cup X_n = U$. An instance I of $R[U]$ satisfies $\bowtie [X_1, \cdots, X_n]$, if $I = \pi_{X_1}(I) \bowtie \cdots \bowtie \pi_{X_n}(I)$.

Consider the relation presented in Table III. Here, the following FD exists: $\{\text{Name}\} \rightarrow \{\text{Disease}\}$. This FD implies that the JD $\bowtie [\{\text{Name}, \text{Doctor}\}, \{\text{Name}, \text{Disease}\}]$ is satisfied, thus lossless decomposition is available (see Table IV).

Let us study the violation of JD constraint in the same table III. The dependency $\bowtie [\{\text{Name}, \text{Doctor}\}, \{\text{Doctor}, \text{Disease}\}]$ is not a valid JD because there is no opportunity to restore the original table. In this case the join of these two tables would lead to appearance of two phantom records, absent in the original table. The key "Brooks" would produce four items instead of two. This is explained by the fact that the dependency $\{\text{Hospital}\} \rightarrow \{\text{Doctor}\}$ is not an FD. Also, there may be the opposite case when the records would be lost.

Currently, several algorithms are being actively developed for efficient testing of existing JDs. For example, the algorithm [18] is aimed for an I/O-efficient testing in the external memory model.

Paper [19] contains the finite axiomatization of the implication problem for inclusion and conditional independence atoms (dependencies) in the dependence logic context. The general axiomatization approach is described, which extend to such types of dependencies as inclusion and join dependencies.

FlightID	Aircraft	Date *(DD/MM/YY)*	Price
1	A320	01.01.17	300
2	A320	04.01.17	340
3	Boeing 747	13.03.17	210
4	A380	15.03.17	410
5	Boeing 747	17.03.17	270
6	A320	23.07.17	450

TABLE VI
DD EXAMPLE

D. Differential Dependencies

Differential dependencies (DD) are the newest concept we review in this paper. They were first presented in 2011 [20] and are defined as follows:

Definition 4: Let $\phi_{LHS}[X]$, $\phi_{RHS}[Y]$ be differential functions specifying constraints on distances over attributes X and Y of relation R. A differential dependency $\phi_{LHS}[X] \rightarrow \phi_{RHS}[Y]$ holds on relation R if for any pair of tuples for which the difference on attributes X satisfies the constraints specified by $\phi_{LHS}[X]$, the difference on Y also satisfies the constraints specified by $\phi_{RHS}[Y]$.

Speaking less formally, we say that if a DD $\phi_{LHS}[X] \rightarrow \phi_{RHS}[Y]$ holds, then for any two tuples on attribute combination X whose distance belongs to the range specified by ϕ_{LHS}, the distance of the same two tuples on Y is in the range specified by ϕ_{RHS}. Differential function on attributes may be specified by operators $\{=, <, >, \leq, \geq\}$. In this case, DD can be reduced to other dependency concepts in the following way: (i) if a differential function for LHS represents a similarity constraint $(= 0)$, then DD becomes matching dependency (MD concept, [21]), (ii) if all differential constraints are $(= 0)$, then DD subsumes FD [20].

Consider an example presented in Figure VI — a table containing flight information. The following DD $[Aircraft(= 0) \wedge Date(\leq 4)] \rightarrow [Price(\geq 40, \leq 60)]$ holds. It means that for the same type of aircraft ($Aircraft(= 0)$) the price difference for flights in any range of four days ($Date(\leq 4)$) must be ≥ 40 and ≤ 60.

If we take away the date from the dependency, the dependency will be violated: $[Aircraft(= 0)] \rightarrow [Price(\geq 40, \leq 60)]$. Let us consider the $A320$ value. There is the following set of $Price$ attribute values corresponding to

$A320$: $\{300, 340, 450\}$. Here, the difference between prices for $FlightID = 1$ and $FlightID = 6$ is outside of the range specified by the dependency. This illustrates the dependency violation.

Since DDs represent a special kind of FD and MD, their application domains overlap. DD has a rather broad application: the data quality problem, integrity constraints checking, and query optimization. Furthermore, the idea of a differential key (a new type of constraint that is based on deduced rules related to attribute distance) provides insight into the data partitioning task.

DD is a more general concept than FD, and its definition is based on distances between attribute values, so FD inference algorithms are not fully capable of DD discovery — they find special cases only. Nevertheless, in the original work [20] DD row- and column-based inference approaches are described.

E. Multivalued Dependencies

Multivalued dependencies (MVD) were introduced independently by R. Fagin and C.A. Zaniolo in 1976 [22], [23]. The modern MVD definition is as follows [24]:

Definition 5: A relation R satisfies the MVD $X \twoheadrightarrow Y$ (where $X, Y \subseteq R$, $X \cap Y = \emptyset$), if and only if for all $t_1,\ t_2 \in R$ holds: if $t_1[XY] = t_2[XY]$ then there is $t \in R$ such that $t[XY] = t_1[XY]$ and $t[X(R - XY)] = t_2[X(R - XY)]$.

That means if we have two tuples of R that agree on X, then their values for Y may be swapped, and the result will be two tuples that are also in the relation R.

In subsequent studies, the MVD definition was slightly modified: LHS and RHS of dependency are no longer necessarily disjoint [25]. A number of MVD inference rules were presented as well.

Traditionally, MVDs have been considered as the necessary and sufficient condition of a relation to be decomposed into two of its projections without loss of information [22]. This means that an MVD $X \twoheadrightarrow Y$ holds if and only if $R = R[XY] \bowtie R[X(R - XY)]$ [24]. Such information on relation decomposability may be used in a schema (re-)design task as it reveals internal structure of a data source.

Lately, MVDs were generalized by bringing in a new concept of full hierarchical dependency, see Section IV-F. Another remarkable fact is the relationship between FDs and MVDs: we may say for $X \cap Y = \emptyset$, that if an FD $X \to Y$ holds in a relation, then an MVD $X \twoheadrightarrow Y$ also holds. Thus, each FD is also an MVD, but not vice versa [25].

Let us consider a relation presented in Figure VII. Attribute $CourseID$ determines a set of values $StudentName$ and $RecommendedBook$. The latter does not depend on attribute $StudentName$ (that means there is no connectivity between these two attributes), and we may say that the MVDs {CourseID} $\twoheadrightarrow$ {StudentName} and {CourseID} $\twoheadrightarrow$ {RecommendedBook} hold in relation.

To show the violation of the dependence, it is sufficient to replace the value of one field in the table VII. The result of the changes is shown in the table VIII, from which we can see that changing the value of attribute RecommendedBook in the third row of the table violates both MVDs. In order to keep the dependencies there should be Course_book_math for Lena and Course_book_optic for Michael.

CourseID	StudentName	RecommendedBook
10	Michael	Course_book_math
10	Michael	Course_book_mechanic
10	Lena	Course_book_math
10	Lena	Course_book_mechanic
106	Michael	Course_book_cs
106	Michael	Course_book_mlearn
9	John	Course_book_reading
9	John	Course_book_grammar

TABLE VII
MVD EXAMPLE

CourseID	StudentName	RecommendedBook
10	Michael	Course_book_math
10	Michael	Course_book_mechanic
10	Lena	**Course_book_optic**
10	Lena	Course_book_mechanic
106	Michael	Course_book_cs
106	Michael	Course_book_mlearn
9	John	Course_book_reading
9	John	Course_book_grammar

TABLE VIII
MVD COUNTEREXAMPLE

For complete understanding of MVDs and their place among other dependency concepts, the following papers are recommended [26], [24].

F. Full Hierarchical Dependencies

As it was already mentioned, a full hierarchical dependency (FHD) is an attempt to generalize the MVD concept. The first notion and formalization of FHD was presented in [27]. Nowadays, the following definition is used [24]:

Definition 6: Let $X \subseteq R$ and S is a non-empty set of pairwise disjoint subsets of R that are also disjoint from X. $S \neq \emptyset$, for all $Y \in S$ we have $Y \subseteq S$ and for all $Y,\ Z \in S \cup \{X\}$ we have $Y \cap Z = \emptyset$. Relation R satisfies FHD $X : Y_1, \ldots, Y_k$, if and only if for all $t_1, \ldots, t_{k+1} \in R$ the following condition is satisfied: if $t_i[X] = t_j[X]$ for all $1 \leq i,\ j \leq k + 1$ then there is some $t \in R$ such that $t[XY_i] = t_i[XY_i]$ for all $i = 1, \ldots, k$ and $t[X(R - XY_1 \ldots Y_k)] = t_{k+1}[X(R - XY_1 \ldots Y_k)]$.

As we can see, in case of $k = 1$ this is the definition of MVD. Moreover, if for each $k = 1, \ldots, n$ MVD $X \twoheadrightarrow Y_k$ holds over R, then R satisfies the FHD $X : \{Y_1, \ldots, Y_n\}$. Therefore, the area of FHD application is the same as the one of MVDs — schema design in terms of relation decomposition. The result of such decomposition is called generalized hierarchical decomposition (GHD), and it may be represented as a tree structure [27].

Since we show in the MVD section that {CourseID} $\twoheadrightarrow$ {StudentName} and {CourseID} $\twoheadrightarrow$ {RecommendedBook} hold, then the FHD {CourseID} : {StudentName, RecommendedBook} also holds.

Category	Weight	Cost	Distance
Portable	$0-5$ kg	150$	$0-10$ km
Portable	$0-5$ kg	200$	$11-20$ km
Portable	$0-5$ kg	250$	$21-30$ km
Midsize	$6-10$ kg	250$	$0-10$ km
Midsize	$6-10$ kg	350$	$11-20$ km
Midsize	$6-10$ kg	450$	$21-30$ km
Largesize	$11-15$ kg	350$	$0-10$ km
Largesize	$11-15$ kg	500$	$11-20$ km
Largesize	$11-15$ kg	650$	$21-30$ km

TABLE IX
OD EXAMPLE

Due to the aforementioned relationships between concepts, FHD mainly appears in studies on combining dependency classes (e.g. [24], where non-trivial FD and FHD connectivity is studied).

G. Order Dependencies

The term "order dependencies" (OD) first appeared in the study [28]. An OD allows to describe the value of some attribute using the information about the order relation on the given value set. For example, we need to transfer an item from point A to point B. OD gives us an opportunity to estimate that in this case delivery by a heavy vehicle will cost at least as much as the delivery by a car as both have to travel the same distance (we ignore real-life aspects of shipping).

The formal definition ODs is as follows [29]:

Definition 7: Call $X \mapsto Y$ an order dependency (where $X, Y \subseteq R$) over the relation R if, for every pair of admissible tuples $t_1, t_2 \in R$, $t_1[X] \preceq t_2[X]$ implies $t_1[Y] \preceq t_2[Y]$.

The analysis of these ODs can be used for improving the efficiency of database query processing. However, arrangement of sets (on which the model of OD is based) is a challenging task. Currently, development of algorithms for fast sets arrangement is a very active area of research. For example, in study [29] an efficient algorithm for lexicographical set arrangement has been introduced. Also, a large number of inference rules that are used for query transformation during the optimization process is presented.

For example, Table IX shows the following OD: {Category, Weight, Distance} $\mapsto$ {Cost}. So, according to the commodity (that is characterized by the weight of the shipment) we can specify whether the delivery costs more or less for the shipping to the same distances. As we have already mentioned, we suppose that portable shipping will cost less than the large size shipping on the same distances.

Consider the dependence {Distance} $\mapsto$ {Cost}. It is not a valid OD since the order relation for the shipment price for different distances will be violated. For example, shipment of $Largesize$ for 0–10 km is cheaper than shipment of $Midsize$ for 21–30 km.

H. Conditional Functional Dependencies

One of the newest FD types is the Conditional Functional Dependencies (CFDs). The first mention of CFD appears in the study by Wenfei Fan et al. [30]. CFDs aim at capturing the consistency of data by enforcing bindings of semantically related values.

SC	NB	City	Client	Rate
110	Bank1	Moscow	Wood	10%
110	Bank2	Bremen	Martin	7%
220	Bank1	Stockholm	King	6%
110	Bank2	St. Petersburg	King	7%
220	Bank1	Hamburg	Turner	6%

TABLE X
CFD EXAMPLE, SOURCE DATA

SC	NB	Rate
220	Bank1	—
110	Bank2	—

TABLE XI
CFD EXAMPLE, PATTERN TABLEU

We frequently encounter CFDs in our everyday life. For example, the position of a worker determines his salary only in some departments but not in the whole organization. CFDs are extensively used in the data integration domain. It is justified by the fact that the patterns existing in the individual data sources would also present in the combined data set.

CFDs extend the FDs using the pattern table that provides the bound of semantically related values. It is important to note that it is necessary to apply CFDs only to tuples that meet the values from the pattern table, but not to all tuples.

The formal definition CFDs is as follows [31]:

Definition 8: A conditional functional dependency (CFD) ϕ on S is a pair $(X \to Y, T_p)$, where $X \to Y$ is a standard FD, referred to as the embedded FD; and T_p is a "pattern tableau" that defines over which rows of the table the embedded FD applies. Each entry $t_p \in T_p$ specifies a pattern over $X \cup Y$, so for each attribute in $A \in X \cup Y$, either $t_p[A] = \alpha$, where α is a value in the domain of A, or else $t_p[A] = \_\_$, for the special wild-card symbol __. A row r_i satisfies an entry t_p of tableau T_p for attributes A, denoted by $r_i[A] \asymp t_p[A]$, if either $r_i[A] = t_p[A]$, or else $t_p[A] = \_\_$. The CFD ϕ holds if

$$\forall i, j, p. r_i[X] = r_j[X] \asymp t_p[X] \Rightarrow r_i[Y] = r_j[Y] \asymp t_p[Y].$$

Consider the Table X that contains client base of various banks. We introduce the following abbreviation: subdivision code – SC, name of the bank – NB. Table X illustrates the following CFD: {Subdivision code, Name of the bank} $\longrightarrow$ {Rate}. The Table XI shows that the banking sector values with an equal subdivision code will probably have an identical interest rate. Although it is important to understand that this condition holds for the majority of tuples in source data, but not for all. For example, Bank2 in St. Petersburg with subdivision code of 110 has above rate than Bank2 in Bremen, which has the same subdivision code.

In order to demonstrate the violation of CFD we can modify Table XI (pattern tableu) in the following way.substitute bank2 entry with the wild-card. In this case values corresponding to $SC = 110$ are not equal: $7, 7, 10$. Thus, the dependency does not hold anymore.

Dependency Concept	Notation	Use-case	Inference Rules and Algorithms
FD	$X \rightarrow Y$	Data cleansing, Query optimization [8]	TANE [32], FDEP [33], DFD [34], FDmine [35]
IND	$X \subseteq Y$	Data integration, Schema design, Integrity checking [36]	Binder [13], Faida [12], Mind [37]
JD	$\bowtie [X_1, \ldots, X_n]$	Schema design [15]	Hu et.al [18]
DD	$\phi_{LHS}[X] \rightarrow \phi_{RHS}[Y]$	Data partition, Query Optimization, Integrity Constraints [20]	Song and Chen [20], Liu et.al [38]
MVD	$X \twoheadrightarrow Y$	Data partition, Schema design [26]	Savnik and Flach [26]
FHD	$X : \{Y_1, \ldots, Y_n\}$	Data partition, Schema design [27]	Biskup and Link [24]
OD	$X \mapsto Y$	Query optimization [28]	Szlichta et.al [29]
CFD	$X \rightarrow Y, T_p$	Consistency checking, Data cleansing [30]	Li et.al [39]

TABLE XII
CUMULATIVE TABLE

An important problem is efficient estimation of CFD confidence using a small number of data passes and a small amount of space. The solution to this problem is described in [31]. An improvement of the algorithm for the minimum CFD set construction has been proposed in the work [39]. The described algorithm works in linear time and is one of the most efficient contemporary algorithms.

V. CONCLUSION

In this paper we surveyed several database dependency concepts. For each type we provided both formal and non-formal definitions and presented an example. We also briefly discussed extraction algorithms and possible use cases. A short summary of considered dependency types is presented in Table XII.

REFERENCES

[1] R. Fagin and M. Y. Vardi, "The theory of data dependencies — an overview," in *Proceedings of the 11th Colloquium on Automata, Languages and Programming*. London, UK, UK: Springer-Verlag, 1984, pp. 1–22. [Online]. Available: http://dl.acm.org/citation.cfm?id=646238.683349

[2] L. Caruccio, V. Deufemia, and G. Polese, "Relaxed functional dependencies — a survey of approaches," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 1, pp. 147–165, Jan 2016.

[3] ——, "On the discovery of relaxed functional dependencies," in *Proceedings of the 20th International Database Engineering & Applications Symposium*, ser. IDEAS '16. New York, NY, USA: ACM, 2016, pp. 53–61. [Online]. Available: http://doi.acm.org/10.1145/2938503.2938519

[4] T. Papenbrock, J. Ehrlich, J. Marten, T. Neubert, J.-P. Rudolph, M. Schönberg, J. Zwiener, and F. Naumann, "Functional dependency discovery: An experimental evaluation of seven algorithms," *Proceedings of the VLDB Endowment*, vol. 8, no. 10, pp. 1082–1093, June 2015. [Online]. Available: http://dx.doi.org/10.14778/2794367.2794377

[5] J. Liu, J. Li, C. Liu, and Y. Chen, "Discover dependencies from data—a review," *IEEE Trans. on Knowl. and Data Eng.*, vol. 24, no. 2, pp. 251–264, Feb. 2012. [Online]. Available: http://dx.doi.org/10.1109/TKDE.2010.197

[6] E. F. Codd, "Further normalization of the data base relational model," *Data base systems*, pp. 33–64, 1972.

[7] H. Darwen and C. Date, "The role of functional dependencies in query decomposition," *Relational Database Writings*, vol. 1991, pp. 133–154, 1989.

[8] G. N. Paulley, "Exploiting functional dependence in query optimization," Ph.D. dissertation, Citeseer, 2001.

[9] W. W. Armstrong, "Dependency structures of data base relationships." in *IFIP congress*, vol. 74. Geneva, Switzerland, 1974, pp. 580–583.

[10] R. Fagin, "A normal form for relational databases that is based on domains and keys," *ACM Trans. Database Syst.*, vol. 6, no. 3, pp. 387–415, Sep. 1981. [Online]. Available: http://doi.acm.org/10.1145/319587.319592

[11] J. M. Smith and D. C. P. Smith, "Database abstractions: Aggregation," *Commun. ACM*, vol. 20, no. 6, pp. 405–413, Jun. 1977. [Online]. Available: http://doi.acm.org/10.1145/359605.359620

[12] K. Sebastian, P. Thorsten, D. Christian, F. Moritz, H. Manuel, Z. Martin, Z. Christian, and N. Felix, "Fast approximate discovery of inclusion dependencies," in *Proceedings of the conference on Database Systems for Business, Technology, and Web (BTW)*, 0 2017.

[13] T. Papenbrock, S. Kruse, J.-A. Quiané-Ruiz, and F. Naumann, "Divide conquer-based inclusion dependency discovery," *Proc. VLDB Endow.*, vol. 8, no. 7, pp. 774–785, Feb. 2015. [Online]. Available: http://dx.doi.org/10.14778/2752939.2752946

[14] A. Koeller and E. A. Rundensteiner, *Heuristic Strategies for the Discovery of Inclusion Dependencies and Other Patterns*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 185–210.

[15] A. V. Aho, C. Beeri, and J. D. Ullman, "The theory of joins in relational databases," *ACM Transactions on Database Systems (TODS)*, vol. 4, no. 3, pp. 297–314, 1979.

[16] J. Rissanen, "Relations with functional and join dependencies and their representation by independent components," *Unpublished manuscript, IBM Research Lab., San Jose, Calif*, 1978.

[17] ——, "Theory of relations for databasesa tutorial survey," in *International Symposium on Mathematical Foundations of Computer Science*. Springer, 1978, pp. 536–551.

[18] X. Hu, M. Qiao, and Y. Tao, "Join dependency testing, loomis-whitney join, and triangle enumeration," in *Proceedings of the 34th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. ACM, 2015, pp. 291–301.

[19] M. Hannula and J. Kontinen, "A finite axiomatization of conditional independence and inclusion dependencies," *Information and Computation*, vol. 249, pp. 121–137, 2016.

[20] S. Song and L. Chen, "Differential dependencies: Reasoning and discovery," *ACM Trans. Database Syst.*, vol. 36, no. 3, pp. 16:1–16:41, Aug. 2011. [Online]. Available: http://doi.acm.org/10.1145/2000824.2000826

[21] W. Fan, "Dependencies revisited for improving data quality," in *Proceedings of the Twenty-seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, ser. PODS '08. New York, NY, USA: ACM, 2008, pp. 159–170. [Online]. Available: http://doi.acm.org/10.1145/1376916.1376940

[22] R. Fagin, "Multivalued dependencies and a new normal form for relational databases," *ACM Trans. Database Syst.*, vol. 2, no. 3, pp. 262–278, Sep. 1977. [Online]. Available: http://doi.acm.org/10.1145/320557.320571

[23] C. A. Zaniolo, "Analysis and design of relational schemata for database systems." Ph.D. dissertation, 1976, aAI7622226.

[24] J. Biskup and S. Link, "Appropriate inferences of data dependencies in relational databases," *Annals of Mathematics and Artificial Intelligence*,

vol. 63, no. 3-4, pp. 213–255, Dec. 2011. [Online]. Available: http://dx.doi.org/10.1007/s10472-012-9275-0

[25] 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*, ser. SIGMOD '77. New York, NY, USA: ACM, 1977, pp. 47–61. [Online]. Available: http://doi.acm.org/10.1145/509404.509414

[26] I. Savnik and P. A. Flach, "Discovery of multivalued dependencies from relations," *Intell. Data Anal.*, vol. 4, no. 3,4, pp. 195–211, Sep. 2000. [Online]. Available: http://dl.acm.org/citation.cfm?id=1294171.1294173

[27] C. Delobel, "Normalization and hierarchical dependencies in the relational data model," *ACM Trans. Database Syst.*, vol. 3, no. 3, pp. 201–222, Sep. 1978. [Online]. Available: http://doi.acm.org/10.1145/320263.320271

[28] S. Ginsburg and R. Hull, "Order dependency in the relational model," *Theoretical computer science*, vol. 26, no. 1-2, pp. 149–195, 1983.

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

[30] W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis, "Conditional functional dependencies for capturing data inconsistencies," *ACM Transactions on Database Systems (TODS)*, vol. 33, no. 2, p. 6, 2008.

[31] G. Cormode, L. Golab, K. Flip, A. McGregor, D. Srivastava, and X. Zhang, "Estimating the confidence of conditional functional dependencies," in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*. ACM, 2009, pp. 469–482.

[32] Y. Huhtala, J. Kärkkäinen, P. Porkka, and H. Toivonen, "Tane: An efficient algorithm for discovering functional and approximate dependencies," *The Computer Journal*, pp. 100–111, 1992.

[33] P. A. Flach and I. Savnik, "Database dependency discovery: A machine learning approach," *AI Commun.*, vol. 12, no. 3, pp. 139–160, Aug. 1999. [Online]. Available: http://dl.acm.org/citation.cfm?id=1216155.1216159

[34] Z. Abedjan, P. Schulze, and F. Naumann, "DFD: Efficient functional dependency discovery," in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, ser. CIKM '14. New York, NY, USA: ACM, 2014, pp. 949–958. [Online]. Available: http://doi.acm.org/10.1145/2661829.2661884

[35] H. Yao, H. J. Hamilton, and C. J. Butz, "Fd_mine: Discovering functional dependencies in a database using equivalences," in *Proceedings of the 2002 IEEE International Conference on Data Mining*, ser. ICDM '02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 729–732. [Online]. Available: http://dl.acm.org/citation.cfm?id=844380.844800

[36] J. Bauckmann, U. Leser, and F. Naumann, *Efficient and exact computation of inclusion dependencies for data integration*, 2010.

[37] F. D. Marchi, S. Lopes, and J.-M. Petit, "Unary and n-ary inclusion dependency discovery in relational databases," *Journal of Intelligent Information Systems*, vol. 32, no. 1, pp. 53–73, 2009. [Online]. Available: http://dx.doi.org/10.1007/s10844-007-0048-x

[38] L. Jixue, K. Selasi, L. Jiuyong, Y. Feiyue, and W. V. Millist, "Discovery of approximate differential dependencies," *CoRR*, vol. abs/1309.3733, 2013. [Online]. Available: http://arxiv.org/abs/1309.3733

[39] J. Li, J. Liu, H. Toivonen, and J. Yong, "Effective pruning for the discovery of conditional functional dependencies," *The Computer Journal*, vol. 56, no. 3, pp. 378–392, 2013.