

# Cleaning Integrated Data: Challenges and Opportunities

Wenfei Fan  
University of Edinburgh & Bell Laboratories  
wenfei@inf.ed.ac.uk

## Abstract

This short paper addresses issues in connection with cleaning data integrated from multiple sources. It advocates the need for a new form of constraints to enforce bindings of semantically related data values, and then proposes a class of such constraints, referred to as conditional constraints. Finally, technical challenges for cleaning data based on these conditional constraints are briefly described.

## 1. Cleaning Integrated Data

One of the central problems in connection with data integration is data cleaning: detect and remove errors, conflicts and inconsistencies from the target data integrated from multiple sources.

Recent statistics reports that dirty data costs US businesses billions of dollars annually (cf. [1]). In our direct experience with Lucent Technologies for providing telecommunication services, errors in data routinely lead to problems like failure to bill for provisioned services, delay in repairing network problems, unnecessary leasing of equipment, and furthermore, loss of revenues, credibility and customers. Enterprises are not the only victims of poor data: it is increasingly common that federal government organizations integrate and use data from various data sources, and dirty data may cause misleading or biased analytical results and decisions.

However important, no practical system is yet in place for effectively cleaning data integrated from multiple sources (see [2] for a recent survey). While a number of tools have been developed to support cleaning (e.g., merge/purge and ETL for data warehouses), a significant portion of the work still needs to be done manually or by low-level programs that are difficult to write and maintain.

When overlapping or redundant information from multiple sources is integrated, inconsistencies or conflicts in the data may emerge as violations of *integrity constraints* on the integrated data. An immediate question is what constraints are needed to specify the consistency of the data? Previous work on constraint-based data cleaning is mostly based on traditional dependencies (e.g., functional and inclusion dependencies – FDs and INDs), which were developed mainly for schema design, but are often insufficient in the context of data cleaning, as illustrated by the example below.

**Example 1.1:** Consider a cust schema, which specifies a customer in terms of the phone (country code (CC), area code (AC), phone number (PN)), name (NM), and address (street (STR), city (CT), zip code (ZIP)). An instance of cust is shown in Fig. 1. Traditional functional dependencies on a cust relation may include:

$$f_1: [CC, AC, PN] \rightarrow [STR, CT, ZIP]$$

$$f_2: [CC, AC] \rightarrow [CT]$$

Traditional FDs are to hold on all the tuples in the relation (indeed they do on Fig. 1). In contrast, the following constraint is supposed to hold only when the country code is 44. That is, for customers in the UK, ZIP determines STR:

$$\phi_0: [CC = 44, ZIP] \rightarrow [STR]$$

In other words,  $\phi_0$  is an FD that is to hold on the subset of tuples that satisfies the pattern “CC = 44”, rather than on the entire cust relation. It is generally *not* considered an FD in the standard definition since  $\phi_0$  includes a *pattern with data values* in its specification.

The following constraints are again not considered FDs:

	CC	AC	PN	NM	STR	CT	ZIP
$t_1$ :	01	908	1111111	Mike	Tree Ave.	NYC	07974
$t_2$ :	01	908	1111111	Rick	Tree Ave.	NYC	07974
$t_3$ :	01	212	2222222	Joe	Elm Str.	NYC	01202
$t_4$ :	01	212	2222222	Jim	Elm Str.	NYC	01202
$t_5$ :	01	215	3333333	Ben	Oak Ave.	PHI	02394
$t_6$ :	44	131	4444444	Ian	High St.	EDI	EH4 1DT

Figure 1: An instance of the cust relation

$$\phi_1: [CC = 01, AC = 908, PN] \rightarrow [STR, CT = MH, ZIP]$$

$$\phi_2: [CC = 01, AC = 212, PN] \rightarrow [STR, CT = NYC, ZIP]$$

$$\phi_3: [CC = 01, AC = 215] \rightarrow [CT = PHI]$$

The first constraint  $\phi_1$  assures that only in the US (country code 01) and for area code 908, if two tuples have the same PN, then they must have the same STR and ZIP values and furthermore, the city *must* be MH. Similarly,  $\phi_2$  assures that if the area code is 212 then the city must be NYC; and  $\phi_3$  specifies that for all tuples in the US and with area code 215, their city must be PHI (irrespective of the values of the other attributes). Observe that  $\phi_1$  and  $\phi_2$  refine the standard FD  $f_1$  given above, while  $\phi_3$  refines the FD  $f_2$ . This refinement essentially enforces a binding of semantically related data values. Note that while tuples  $t_1$  and  $t_2$  in Fig. 1 do not violate  $f_1$ , they violate its refined version  $\phi_1$ , since the city cannot be NYC if the area code is 908.  $\square$

The constraints given above are conditional: they are required to hold on tuples of certain patterns. Constraints that hold conditionally may arise in a number of domains. In particular, dependencies that apply conditionally appear to be needed when integrating data, since dependencies that hold only in a subset of sources will hold only conditionally in the integrated data.

However desirable, these constraints cannot be expressed as standard FDs or other traditional dependencies. In light of this it is necessary to develop a new form of constraints that are capable of capturing the notion of “correct data” in these situations.

## 2. Conditional Constraints

We propose an extension of traditional functional and inclusion dependencies, referred to as *conditional functional* and *conditional inclusion dependencies* (CFDs and CINDs), respectively, that are able to express constraints that hold only conditionally while retaining the simplicity of their traditional counterparts.

Consider a relation schema  $R$  defined over a fixed set of attributes, denoted by  $\text{attr}(R)$ .

**CFDs.** A CFD  $\varphi$  on  $R$  is a pair  $(R : X \rightarrow Y, T_p)$ , where (1)  $X, Y$  are sets of attributes from  $\text{attr}(R)$ , (2)  $R : X \rightarrow Y$  is a standard FD, referred to as the FD *embedded in*  $\varphi$ ; and (3)  $T_p$  is a tableau with all attributes in  $X$  and  $Y$ , referred to as the *pattern tableau* of  $\varphi$ , where for each  $A$  in  $X$  or  $Y$  and each tuple  $t \in T_p$ ,  $t[A]$  is either a constant ‘a’ in the domain of  $A$ , or an unnamed variable ‘.’. If  $A$  appears in both  $X$  and  $Y$ , we use  $t[A_l]$  and  $t[A_r]$  to indicate the  $A$  field of  $t$  corresponding to  $A$  in  $X$  and  $Y$ , respectively.

**Example 2.1:** The constraints  $\phi_0, f_1, \phi_1, \phi_2, f_2, \phi_3$  on cust given in Example 1.1 can be expressed as CFDs  $\varphi_1$  (for  $\phi_0$ ),  $\varphi_2$  (for  $f_1, \phi_1$  and  $\phi_2$ , one per line) and  $\varphi_3$  (for  $f_2, \phi_3$ ), as shown in Fig. 2.  $\square$

(a) Tableau  $T_1$  of  $\varphi_1 = (\text{cust}:[\text{CC}, \text{ZIP}] \rightarrow [\text{STR}], T_1)$

CC	ZIP	STR
44	-	-

(b) Tableau  $T_2$  of  $\varphi_2 = ([\text{CC}, \text{AC}, \text{PN}] \rightarrow [\text{STR}, \text{CT}, \text{ZIP}], T_2)$

CC	AC	PN	STR	CT	ZIP
01	908	-	-	MH	-
01	212	-	-	NYC	-

(c) Tableau  $T_3$  of  $\varphi_3 = ([\text{CC}, \text{AC}] \rightarrow [\text{CT}], T_3)$

CC	AC	CT
01	215	PHI

**Figure 2: Example CFDs**

Intuitively, the pattern tableau  $T_p$  of  $\varphi$  refines the standard FD embedded in  $\varphi$  by enforcing the binding of semantically related data values. To define the semantics of  $\varphi$ , we first introduce a notation. Given two tuples  $t_1, t_2$  and an attribute  $A$ , we say that  $t_1[A]$  and  $t_2[A]$  *match*, denoted by  $t_1[A] \asymp t_2[A]$ , if either  $t_1[A]$  or  $t_2[A]$  is ‘-’, or both  $t_1[A]$  and  $t_2[A]$  are the same data constant. This notion can be readily extended to tuples. For example,  $t[A, B] = (a, b) \asymp t_c[A, B] = (a, \_)$ .

A relation  $I$  of  $R$  satisfies the CFD  $\varphi$  if for *each pair* of tuples  $t_1, t_2$  in  $I$ , and for *each* tuple  $t_c$  in the pattern tableau  $T_p$  of  $\varphi$ , if  $t_1[X] = t_2[X] \asymp t_c[X]$ , then  $t_1[Y] = t_2[Y] \asymp t_c[Y]$ . That is, if  $t_1[X]$  and  $t_2[X]$  are equal and in addition, they both match the pattern  $t_c[X]$ , then  $t_1[Y]$  and  $t_2[Y]$  must also be equal to each other and match the pattern  $t_c[Y]$ .

**Example 2.2:** The cust relation in Fig. 1 satisfies  $\varphi_1$  and  $\varphi_3$  of Fig. 2. However, it does not satisfy  $\varphi_2$ . Indeed, tuple  $t_1$  violates the pattern tuple  $t_c = (01, 908, \_, \_, \text{MH}, \_)$  in tableau  $T_2$  of  $\varphi_2$ :  $t_1[\text{CC}, \text{AC}, \text{PN}] = t_1[\text{CC}, \text{AC}, \text{PN}] \asymp (01, 908, \_)$ , but  $t_1[\text{STR}, \text{CT}, \text{ZIP}] \not\asymp (\_, \text{MH}, \_)$  since  $t_1[\text{CT}]$  is NYC rather than MH; similarly for  $t_2$ .  $\square$

A standard FD  $X \rightarrow Y$  can be expressed as a CFD ( $X \rightarrow Y, T_p$ ) in which  $T_p$  contains a single tuple consisting of ‘-’ only.

**CINDs.** A CIND  $\psi$  is a pair  $(R_1[X; Z_X] \subseteq R_2[Y; Z_Y], T_p)$ , where (1)  $X, Z_X$  and  $Y, Z_Y$  are lists of attributes from  $\text{attr}(R_1)$  and  $\text{attr}(R_2)$ , respectively, (2)  $R_1[Z_X] \subseteq R_2[Z_Y]$  is a standard IND, referred to as the *embedded* IND; and (3)  $T_p$  is a pattern tableau, which contains all attributes in  $X, Z_X$  and  $Y, Z_Y$ .

An instance  $(I_1, I_2)$  of  $(R_1, R_2)$  satisfies the CIND  $\psi$  if for all  $t_1 \in I_1$  and all pattern  $t_p \in T_p$ , if  $t_1[X, Z_X] \asymp t_p[X, Z_X]$ , then there exists  $t_2 \in I_2$  such that  $t_2[Y, Z_Y] \asymp t_p[Y, Z_Y]$  and moreover,  $t_2[Z_Y] = t_1[Z_X]$ . That is, if  $t_1[X, Z_X]$  matches a pattern  $t_p[X, Z_X]$ , then there must be a  $t_2 \in I_2$  such that  $t_2[Y, Z_Y]$  matches  $t_p[Y, Z_Y]$  and moreover,  $t_1[Z_X]$  and  $t_2[Z_Y]$  are equal.

A standard IND  $R_1[Z_X] \subseteq R_2[Y]$  can be expressed as a CIND  $(R_1[\emptyset; Z_X] \subseteq R_2[\emptyset; Z_Y], T_p)$ , in which  $T_p$  contains a single tuple consisting of ‘-’ only.

**Example 2.3:** Consider an equip schema, which catalogs equipment installed for US customers (identified by AC, PN) and includes manufacturer (eqmfct), model number (eqmodel), and the serial number. Assume that for US customers (CC = 01), if their area code is 908 (AC = 908), then they use equipment manufactured by LU with model number ZE400. This can be expressed as a CIND ( $\text{cust}[X; Z_X] \subseteq \text{equip}[Y; Z_Y], T_p$ ), in which  $T_p$  contains a pattern tuple  $t_p$  such that  $t_p[X] = t_p[\text{CC}] = 01$ ,  $t_p[Z_X] = t_p[\text{AC}, \text{PN}] = (908, \_)$ ,  $t_p[Y] = t_p[\text{eqmfct}, \text{eqmodel}] = (\text{LU}, \text{ZE400})$ , and  $t_p[Z_Y] = t_p[\text{AC}, \text{PN}] = (908, \_)$ . The tuple  $t_p$  indicates that the

constraint is applicable if CC = 01 and AC = 908. Note that neither CC is an attribute in the equip table, and nor eqmfct, eqmodel are attributes of cust.  $\square$

### 3. Challenges and Opportunities

In contrast to their traditional counterparts studied for schema design, conditional constraints aim at improving the quality of data and are more helpful in data cleaning. However, already a difficult problem when traditional FDs and INDs are considered, data cleaning based on CFDs and CINDs is even more challenging. Below we describe a few open problems that require serious research efforts.

First, to clean data with conditional constraints, the constraints must be *consistent*, i.e. there must exist a nonempty database that satisfies the constraints. As opposed to standard FDs and INDs, a set of CFDs and CINDs may be inconsistent. As an example, consider CFD  $\psi_1 = (R : [A] \rightarrow [B], T_1)$ , where  $T_1$  consists of two pattern tuples  $(\_, b)$  and  $(\_, c)$ . Then no nonempty instance  $I$  of  $R$  can possibly satisfy  $\psi_1$ . Indeed, for any tuple  $t$  in  $I$ , while the first pattern tuple says that  $t[B]$  must be  $b$  no matter what value  $t[A]$  has, the second pattern requires  $t[B]$  to be  $c$ . With this comes the need for studying the *consistency* problem, the problem for determining whether a set of CFDs and CINDs is consistent. This problem is NP-complete for CFDs alone, and is PSPACE-complete for CFDs and CINDs. Thus efficient approximate or heuristic algorithms have to be in place for checking the consistency of conditional constraints, which are, however, nontrivial to develop.

Second, in order to clean (integrated) data based on conditional constraints, one has to identify violations of the constraints. Constraint validation may be expensive when the data is large. This calls for algorithms that automatically generate efficient queries to find violations of conditional constraints. Moreover, when the data is changed in the cleaning (repairing) process, we need techniques for incrementally computing violation of the constraints in response to updates to the data. These are also nontrivial.

Third, in practice one often wants to automatically resolve the violations. This highlights the need for computing a *repair*  $I'$  of dirty data  $I$  by updating  $I$ , such that (a)  $I'$  is also an instance of the target schema, (b)  $I'$  is consistent, i.e. it satisfies the constraints, and (c) the updates on  $I$  to obtain  $I'$  are *accurate* and *minimal*. The main challenges here include the development of an accurate model for data repairing, and an efficient algorithm for finding a repair. The repair model may need to incorporate the analysis of data provenance. For finding a repair, it is already NP-complete for standard FDs or INDs alone (data complexity), in any reasonable repair model. It is not easy to develop efficient approximate/heuristic algorithms to find database repair based on CFDs and CINDs.

Fourth, data cleaning should be seamlessly incorporated into the data integration process [2]. This introduces a rich set of problems, including *constraint propagation* and *feedback loops*. The former is to derive, given an integration mapping  $\sigma$  and a set  $\Sigma$  of CFDs and CINDs on the sources, the set  $\Sigma_T$  of CFDs and CINDs *propagated* from  $\Sigma$  via  $\sigma$  on the target data, such that for data sources  $I$ , if they satisfy  $\Sigma$ , then the target data  $\sigma(I)$  satisfies  $\Sigma_T$ . This is important for discovering CFDs and CINDs on the target data. The latter problem is to propagate changes made on the target data in the repairing process back to the sources. These problems are nontrivial.

### 4. References

- [1] W. W. Eckerson. Data Quality and the Bottom Line: Achieving Business Success through a Commitment to High Quality Data. Technical report, The Data Warehousing Institute, 2002.
- [2] E. Rahm and H. H. Do. Data cleaning: Problems and current approaches. *IEEE Data Engineering Bulletin*, 23(4), 2000.