Data Processing and Analytics (DISS-DPA)

Principles of Data Quality – Cleaning with Constraints

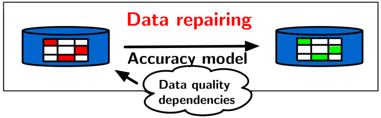
Christopher Spinrath

Database Group (BD) – CNRS – LIRIS – Université Lyon 1

Fall 2025

This presentation is based on slides by Angela Bonifati

Objectives



Objectives

- Criteria for data quality
- Data quality dependencies and why we want them
- Key problems and algorithmic challenges
- Data improvement dependencies
- Repair models
- The chase, and why and how it is extended for repairing
- Strategies for resolving conflicts

Outline

1. The Data Quality Problem
2. Conditional Dependencies for Data Consistency
3. Matching Dependencies for Record Matching
4. Other Kinds of Dependencies
5. Key Algorithmic Challenges

The Data Quality Problem

A Real-World Encounter

“Mr. Smith, our database records indicate that you owe us an outstanding amount of £1,921.76 for council tax in 2007”

A Real-World Encounter

"Mr. Smith, our database records indicate that you owe us an outstanding amount of £1,921.76 for council tax in 2007"

A Data Quality Problem

- Mr. Smith moved from Edinburgh to London in 2006, and no longer lived in Edinburgh in 2007
- The council database was not correctly updated: it retains both Smith's old and new address

NI#	AC	phn	name	street	city	zip
SC1234566	131	1234567	M. Smith	Mayfield	EDI	EH4 8LE
SC1234566	020	1234567	M. Smith	Portland	LDN	W1B 1JL

A Real-World Encounter

"Mr. Smith, our database records indicate that you owe us an outstanding amount of £1,921.76 for council tax in 2007"

A Data Quality Problem

- Mr. Smith moved from Edinburgh to London in 2006, and no longer lived in Edinburgh in 2007
- The council database was not correctly updated: it retains both Smith's old and new address

NI#	AC	phn	name	street	city	zip
SC1234566	131	1234567	M. Smith	Mayfield	EDI	EH4 8LE
SC1234566	020	1234567	M. Smith	Portland	LDN	W1B 1JL

Statistics

50% of bills have errors (phone bill reviews, 1992)

Customer Data

Example (Customer Data)

country	AC	phn	street	city	zip
44	131	1234567	Mayfield	New York	EH8 9LE
44	131	3456789	Crichton	New York	EH8 9LE
01	908	3456789	Mountain Ave	New York	07974

Anything wrong?

Customer Data

Example (Customer Data)

country	AC	phn	street	city	zip
44	131	1234567	Mayfield	New York	EH8 9LE
44	131	3456789	Crichton	New York	EH8 9LE
01	908	3456789	Mountain Ave	New York	07974

Anything wrong?

Errors

- New York City is not in the UK which has country code 44
- Murray Hill, which has country/area code 01/908, is not in New York (but in New Jersey)

Customer Data

Example (Customer Data)

country	AC	phn	street	city	zip
44	131	1234567	Mayfield	New York	EH8 9LE
44	131	3456789	Crichton	New York	EH8 9LE
01	908	3456789	Mountain Ave	New York	07974

Anything wrong?

Errors

- New York City is not in the UK which has country code 44
- Murray Hill, which has country/area code 01/908, is not in New York (but in New Jersey)

Statistics

Customer records have error rates of 10% – 75% (telecommunication)

Real-World Data is Often Dirty

Dirty Data

Data that is inconsistent, inaccurate, incomplete, stale, or deliberately falsified

Real-World Data is Often Dirty

Dirty Data

Data that is inconsistent, inaccurate, incomplete, stale, or deliberately falsified

Examples

- The Pentagon asked over 200 dead officers to re-enlist
- In the UK there are 81 million national insurance numbers but only 60 million people eligible
- 500,000 dead people in Australia retain active medicare cards
- In a database of 500,000 customers, 120,000 records become invalid within a year – death, divorce, marriage, move

Real-World Data is Often Dirty

Dirty Data

Data that is inconsistent, inaccurate, incomplete, stale, or deliberately falsified

How does Data Get Dirty?

Errors and inconsistencies may be introduced during data gathering, storage, transmission, transformation, integration, ...

Examples

- The Pentagon asked over 200 dead officers to re-enlist
- In the UK there are 81 million national insurance numbers but only 60 million people eligible
- 500,000 dead people in Australia retain active medicare cards
- In a database of 500,000 customers, 120,000 records become invalid within a year – death, divorce, marriage, move

Dirty Data is Costly

Example (Telecommunication Services)

Dirty data routinely leads to

- failure to bill for services,
- delay in repairing network problems, and
- unnecessary leasing of equipment

and consequently to

- loss of revenue, credibility, and customers

Dirty Data is Costly

Example (Telecommunication Services)

Dirty data routinely leads to

- failure to bill for services,
- delay in repairing network problems, and
- unnecessary leasing of equipment

and consequently to

- loss of revenue, credibility, and customers

Examples

- Poor data costs US companies $600 billions annually
- Erroneously priced data in retail databases costs US customers $2.5 billion each year
- World-wide losses from payment card fraud reached $4.84 billion in 2006
- 30% – 80% of the development time for data cleaning in a data integration project

This is true also in finance, life science, e-government, ...

Dirty Data is Costly

Example (Telecommunication Services)

Dirty data routinely leads to

- failure to bill for services,
- delay in repairing network problems, and
- unnecessary leasing of equipment

and consequently to

- loss of revenue, credibility, and customers

Examples

- Poor data costs US companies $600 billions annually
- Erroneously priced data in retail databases costs US customers $2.5 billion each year
- World-wide losses from payment card fraud reached $4.84 billion in 2006
- 30% – 80% of the development time for data cleaning in a data integration project

This is true also in finance, life science, e-government, ...

The N° 1 Problem

Data quality: The N° 1 problem for data management!

The Need for Data Quality Tools

Manual Effort

is beyond reach in practice

- For instance, editing a sample of census data easily took dozens of clerks several months (Winkler 04, US Census Bureau)

The Need for Data Quality Tools

Manual Effort

is beyond reach in practice

- For instance, editing a sample of census data easily took dozens of clerks several months (Winkler 04, US Census Bureau)

Data Quality Tools

Help to automatically

- discover data quality rules
- reason about these rules
- detect errors based on violations of the these rules
- repair (or suggest repairs) of data

and this in a principled way

Existing Tools

ETL

Most data quality tools adhere to ETL

- E Extraction: Data is collected from sources
- T Transformation: Rules and functions are applied on the data
- L Loading: Results are loaded into the customer's database (warehouse)

Existing Tools

- Are often domain specific, e.g. for addresses
- Transformation rules are manually designed
- Low-level programs

There are many good systems and prototypes around, e.g, AJAX, Potter's Wheel, Usher, Guided Data Repair, ...

Our Goal

Is to complement existing tools by providing a uniform approach to several data quality tasks

What is Data Quality? Some Criteria

Consistency

Whether the data contains errors or conflicts that emerge as violations of certain semantic rules

Example

A patient has age 82 and age 20

What is Data Quality? Some Criteria

Consistency

Whether the data contains errors or conflicts that emerge as violations of certain semantic rules

Example

A patient has age 82 and age 20

Accuracy

How close a value representing a real-life entity is to the true value of the entity

Example

The age of high school students is at most 40 vs. exactly 15

What is Data Quality? Some Criteria

Consistency

Whether the data contains errors or conflicts that emerge as violations of certain semantic rules

Example

A patient has age 82 and age 20

Accuracy

How close a value representing a real-life entity is to the true value of the entity

Example

The age of high school students is at most 40 vs. exactly 15

Completeness

Whether a given query can be answered given the information available

Example

A missing value (the age of a patient is null), or missing tuples (no entry for a patient)

What is Data Quality? Some Criteria

Consistency

Whether the data contains errors or conflicts that emerge as violations of certain semantic rules

Example

A patient has age 82 and age 20

Accuracy

How close a value representing a real-life entity is to the true value of the entity

Example

The age of high school students is at most 40 vs. exactly 15

Completeness

Whether a given query can be answered given the information available

Example

A missing value (the age of a patient is null), or missing tuples (no entry for a patient)

Timeliness

Whether the data is too stale to answer a query, or what is the most recent value?

Example

Council tax collection in 2007 is based on an old address of 2005

Constraint-Based Data Cleaning

Goal

We want various integrity constraint formalisms to help us achieve a fundamental approach for improving the quality of data

Goal

We want various integrity constraint formalisms to help us achieve a fundamental approach for improving the quality of data

The Plan

- As a warm-up, we illustrate this with standard dependency classes (functional and inclusion dependencies)
- We argue that they have to be extended to accommodate for some of the data quality criteria
- We present various classes of quality dependencies

Reminder – Schemas and Database Instances

Schemas

- A relation schema consists of
  - a name, used to identify the relation schema
  - a set of attributes $\{A_1, \ldots, A_m\}$
- A (database) schema is a set of relation schemas with pairwise different names

Reminder – Schemas and Database Instances

Schemas

- A relation schema consists of
  - a name, used to identify the relation schema
  - a set of attributes $\{A_1, \ldots, A_m\}$
- A (database) schema is a set of relation schemas with pairwise different names

Example (Schema)

A schema consisting of two relation schemas:

1. **Address** with attributes street, city, and zip
2. **Employee** with attributes name, and department

Reminder – Schemas and Database Instances

Schemas

- A relation schema consists of
  - a name, used to identify the relation schema
  - a set of attributes $\{A_1, \ldots, A_m\}$
- A (database) schema is a set of relation schemas with pairwise different names

Example (Schema)

A schema consisting of two relation schemas:

1. **Address** with attributes street, city, and zip
2. **Employee** with attributes name, and department

Database Instances

- A database instance of a database schema contains, for each relation schema R in it
  - a relation (or table) R consisting of tuples (or rows)
  - with the attributes required by the schema

Reminder – Schemas and Database Instances

Schemas

- A relation schema consists of
  - a name, used to identify the relation schema
  - a set of attributes $\{A_1, \ldots, A_m\}$
- A (database) schema is a set of relation schemas with pairwise different names

Database Instances

- A database instance of a database schema contains, for each relation schema R in it
  - a relation (or table) R consisting of tuples (or rows)
  - with the attributes required by the schema

Example (Schema)

A schema consisting of two relation schemas:

1. **Address** with attributes street, city, and zip
2. **Employee** with attributes name, and department

Example (Database Instance)

Relation Address

street	city	zip
Mayfield	EDI	EH4 8LE
Portland	LDN	W1B 1JL

Relation Employee

name	department
M. Smith	IT

Reminder – Functional Dependencies

Functional Dependencies (FDs)

A functional dependency over a relation schema R has the form

$$R[A_1, \dots, A_m \to B_1, \dots, B_\ell]$$

where $A_1, \dots, A_m, B_1, \dots, B_\ell$ are attributes of R

Reminder – Functional Dependencies

Functional Dependencies (FDs)

A functional dependency over a relation schema R has the form

$$R[A_1, \dots, A_m \to B_1, \dots, B_\ell]$$

where $A_1, \dots, A_m, B_1, \dots, B_\ell$ are attributes of R

Semantics

Let D be a database instance containing a relation R
We say that D satisfies a FD $R[A_1, \dots, A_m \to B_1, \dots, B_\ell]$ if

- whenever two tuples of R agree on the values of $A_1, \dots, A_m$,
- then they also agree on the value of $B_1, \dots, B_\ell$
- Notation: $D \models R[A_1, \dots, A_m \to B_1, \dots, B_\ell]$

Example – Functional Dependency

Example (Functional Dependencies)

$$\text{Customer}[\text{NI\#} \rightarrow \text{name, AC, phn, street, city, zip}]$$

- NI# is a key: there is a single record for each distinct NI#

Database Instance

Relation Customer

NI#	AC	phn	name	street	city	zip
SC1234566	131	1234567	M. Smith	Mayfield	EDI	EH4 8LE
SC1234566	020	1234567	M. Smith	Portland	LDN	W1B 1JL

Example – Functional Dependency

Example (Functional Dependencies)

Customer[NI# $\rightarrow$ name, AC, phn, street, city, zip]

- NI# is a key: there is a single record for each distinct NI#

Database Instance

Relation Customer

NI#	AC	phn	name	street	city	zip
SC1234566	131	1234567	M. Smith	Mayfield	EDI	EH4 8LE
SC1234566	020	1234567	M. Smith	Portland	LDN	W1B 1JL

- The database instance does not satisfy the FD
- For SC1234566, at least one of the records must be dirty

Example – Functional Dependency

Example (Functional Dependencies)

Customer[NI# $\rightarrow$ name, AC, phn, street, city, zip]

- NI# is a key: there is a single record for each distinct NI#

Database Instance

Relation Customer

NI#	AC	phn	name	street	city	zip
SC1234566	131	1234567	M. Smith	Mayfield	EDI	EH4 8LE
SC1234566	020	1234567	M. Smith	Portland	LDN	W1B 1JL

Error Detection

Functional dependencies help to detect errors within a **single** relation

Reminder – Inclusion Dependencies

Inclusion Dependencies (INDs)

An inclusion dependency over relation schemas R and S has the form

$$R[A_1, \dots, A_m] \subseteq S[B_1, \dots, B_m]$$

where $A_1, \dots, A_m$ are attributes of R and $B_1, \dots, B_m$ are attributes of S

Reminder – Inclusion Dependencies

Inclusion Dependencies (INDs)

An inclusion dependency over relation schemas R and S has the form

$$R[A_1, \dots, A_m] \subseteq S[B_1, \dots, B_m]$$

where $A_1, \dots, A_m$ are attributes of R and $B_1, \dots, B_m$ are attributes of S

Semantics

Let D be a database instance containing relations R and S
We say that D satisfies an IND $R[A_1, \dots, A_m] \subseteq S[B_1, \dots, B_m]$ if

- for every tuples t_1 in R,
- there is a tuple t_2 in S
- such that $t_2[B_1, \dots, B_m] = t_1[A_1, \dots, A_m]$
- Notation: $(R, S) \models R[A_1, \dots, A_m] \subseteq S[B_1, \dots, B_\ell]$ or $D \models R[A_1, \dots, A_m] \subseteq S[B_1, \dots, B_m]$

Example – Inclusion Dependencies

Example (Inclusion Dependency)

$$\text{Book}[\text{asin, title, price}] \subseteq \text{Item}[\text{asin, title, price}]$$

Every book sold by a store must be an item carried by the store

Example – Inclusion Dependencies

Example (Inclusion Dependency)

$$\text{Book}[\text{asin, title, price}] \subseteq \text{Item}[\text{asin, title, price}]$$

Every book sold by a store must be an item carried by the store

Database Instance

Relation Book

asin	isbn	title	price
a23	b32	Le Petit Prince	17,99€
a56	b65	Snow White	7,94€

Relation Item

asin	title	type	price
a23	Le Petit Prince	book	17,99€
a12	John Denver	CD	7,94€

Example – Inclusion Dependencies

Example (Inclusion Dependency)

$$\text{Book}[\text{asin}, \text{title}, \text{price}] \subseteq \text{Item}[\text{asin}, \text{title}, \text{price}]$$

Every book sold by a store must be an item carried by the store

- The database instance does not satisfy the inclusion dependency
- The book with asin a56 does not have a counter part in the item relation

Database Instance

Relation Book

asin	isbn	title	price
a23	b32	Le Petit Prince	17,99€
a56	b65	Snow White	7,94€

Relation Item

asin	title	type	price
a23	Le Petit Prince	book	17,99€
a12	John Denver	CD	7,94€

Example – Inclusion Dependencies

Example (Inclusion Dependency)

$$\text{Book}[\text{asin}, \text{title}, \text{price}] \subseteq \text{Item}[\text{asin}, \text{title}, \text{price}]$$

Every book sold by a store must be an item carried by the store

- The database instance does not satisfy the inclusion dependency
- The book with asin a56 does not have a counter part in the item relation

Database Instance

Relation Book

asin	isbn	title	price
a23	b32	Le Petit Prince	17,99€
a56	b65	Snow White	7,94€

Relation Item

asin	title	type	price
a23	Le Petit Prince	book	17,99€
a12	John Denver	CD	7,94€

Error Detection

Inclusion dependencies help to detect errors **across** relations

More Reasons for Using Dependencies

Why do we use Dependencies?

- They capture a fundamental part of the semantics of data
  - Errors and inconsistencies manifest as violations of dependencies
- Techniques and inference systems are in place for reasoning about dependencies
  - removal of redundant dependencies
  - identification of dirty dependencies
- Various algorithms exist to discover dependencies from sample data
- Repair algorithms, based on the so-called chase procedure, are studied in depth

More Reasons for Using Dependencies

Why do we use Dependencies?

- They capture a fundamental part of the semantics of data
  - Errors and inconsistencies manifest as violations of dependencies
- Techniques and inference systems are in place for reasoning about dependencies
  - removal of redundant dependencies
  - identification of dirty dependencies
- Various algorithms exist to discover dependencies from sample data
- Repair algorithms, based on the so-called chase procedure, are studied in depth

Claim

Dependencies should become part of data cleaning processes

Conditional Dependencies for Data Consistency

Example

Relation Address

CC	AC	phn	name	street	city	zip
44	131	1234567	Mike	Mayfield	NYC	EH4 8LE
44	131	3456789	Sarah	Crichton	NYC	EH4 8LE
01	908	3456789	Alex	Mtn Ave	NYC	07974

Functional Dependencies

- Address[CC, AC, phn $\rightarrow$ street]
- Address[CC, AC $\rightarrow$ city, zip]

Revising Traditional Constraints

Example

Relation Address

CC	AC	phn	name	street	city	zip
44	131	1234567	Mike	Mayfield	NYC	EH4 8LE
44	131	3456789	Sarah	Crichton	NYC	EH4 8LE
01	908	3456789	Alex	Mtn Ave	NYC	07974

Functional Dependencies

- Address[CC, AC, phn $\rightarrow$ street]
- Address[CC, AC $\rightarrow$ city, zip]

- The database instance satisfies the FDs
- But the data is not clean!

Observations

- Traditional constraints were designed for improving the quality of relational schemas
- But we want constraints for improving the quality of data

Revising Traditional Constraints

Example

Relation Address

CC	AC	phn	name	street	city	zip
44	131	1234567	Mike	Mayfield	NYC	EH4 8LE
44	131	3456789	Sarah	Crichton	NYC	EH4 8LE
01	908	3456789	Alex	Mtn Ave	NYC	07974

Functional Dependencies

- Address[CC, AC, phn $\rightarrow$ street]
- Address[CC, AC $\rightarrow$ city, zip]

Semantic Properties

This instance is not clean since we know the following semantic properties

- "In the UK, the zip code uniquely determines the street"
- "In the USA, if the area code is 908, then the city must be Murray Hill (MH)"
- "In the UK, if the area code is 131, then the city must be Edinburgh (EDI)"

Revising Traditional Constraints

Example

Relation Address

CC	AC	phn	name	street	city	zip
44	131	1234567	Mike	Mayfield	NYC	EH4 8LE
44	131	3456789	Sarah	Crichton	NYC	EH4 8LE
01	908	3456789	Alex	Mtn Ave	NYC	07974

Functional Dependencies

- Address[CC, AC, phn $\rightarrow$ street]
- Address[CC, AC $\rightarrow$ city, zip]

Semantic Properties

This instance is not clean since we know the following semantic properties

- "In the UK, the zip code uniquely determines the street"
- "In the USA, if the area code is 908, then the city must be Murray Hill (MH)"
- "In the UK, if the area code is 131, then the city must be Edinburgh (EDI)"

These properties cannot be enforced by standard FDs. How can we extend FDs (minimally)?

FDs as First-Order Sentences

A functional dependency

$$R[A_1, \ldots, A_m \to B]$$

can be written as

$$\forall t_1 \forall t_2 \Big(\big(R(t_1) \wedge R(t_2) \wedge \bigwedge_{i \in [1,m]} t_1[A_i] = t_2[A_i] \big) \to t_1[B] = t_2[B] \Big)$$

Reminder – FDs as First-Order Sentences

FDs as First-Order Sentences

A functional dependency

$$R[A_1, \ldots, A_m \to B]$$

can be written as

$$\forall t_1 \forall t_2 \Big(\big(R(t_1) \wedge R(t_2) \wedge \bigwedge_{i \in [1,m]} t_1[A_i] = t_2[A_i] \big) \to t_1[B] = t_2[B] \Big)$$

Idea

To express the previous semantic properties in a similar formalism we add equality with constants

Conditional Functional Dependencies (CFDs)[1]

Example (Conditional Functional Dependency (CFD))

"In the UK, the zip code uniquely determines the street"

$$\forall t_1 \forall t_2 \Big(\big(\mathsf{Address}(t_1) \wedge \mathsf{Address}(t_2) \wedge$$
$$t_1[\mathsf{zip}] = t_2[\mathsf{zip}] \wedge t_1[\mathsf{CC}] = t_2[\mathsf{CC}] \wedge t_1[\mathsf{CC}] = 44\big) \rightarrow t_1[\mathsf{street}] = t_2[\mathsf{street}] \Big)$$

[1] Fan, "Dependencies revisited for improving data quality", *Proceedings of the Twenty-Seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS*,. 2008

Conditional Functional Dependencies (CFDs)[1]

Example (Conditional Functional Dependency (CFD))

"In the UK, the zip code uniquely determines the street"

$$\forall t_1 \forall t_2 \Big(\big(\mathsf{Address}(t_1) \land \mathsf{Address}(t_2) \land$$
$$t_1[\mathsf{zip}] = t_2[\mathsf{zip}] \land t_1[\mathsf{CC}] = t_2[\mathsf{CC}] \land t_1[\mathsf{CC}] = 44\big) \rightarrow t_1[\mathsf{street}] = t_2[\mathsf{street}] \Big)$$

Compact Notation

$\mathsf{Address}[\mathsf{CC} = 44, \mathsf{zip} \rightarrow \mathsf{street}]$

[1] Fan, "Dependencies revisited for improving data quality", *Proceedings of the Twenty-Seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS*,. 2008

Conditional Functional Dependencies (CFDs)[1]

Example (Conditional Functional Dependency (CFD))

"In the UK, the zip code uniquely determines the street"

$$\forall t_1 \forall t_2 \Big(\big(\text{Address}(t_1) \wedge \text{Address}(t_2) \wedge \\ t_1[\text{zip}] = t_2[\text{zip}] \wedge t_1[\text{CC}] = t_2[\text{CC}] \wedge t_1[\text{CC}] = 44\big) \rightarrow t_1[\text{street}] = t_2[\text{street}] \Big)$$

Compact Notation

$\text{Address}[\text{CC} = 44, \text{zip} \rightarrow \text{street}]$

Observations

- It is a **conditional** FD: it may not hold for other countries, e.g., for France
- There is no equivalent standard FD: it cannot be expressed without constants

[1] Fan, "Dependencies revisited for improving data quality", *Proceedings of the Twenty-Seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS*,. 2008

Conditional Functional Dependencies (CFDs)

Example (Conditional Functional Dependency (CFD))

"In the UK, if the area code is 131, then the city must be Edinburgh (EDI)"

$$\begin{aligned}\forall t_1 \forall t_2 \Big(& \big(\text{Address}(t_1) \wedge \text{Address}(t_2) \wedge \\ & t_1[\text{CC}] = t_2[\text{CC}] \wedge t_1[\text{AC}] = t_2[\text{AC}] \wedge t_1[\text{CC}] = 44 \wedge t_1[\text{AC}] = 131\big) \\ & \rightarrow \big(t_1[\text{city}] = t_2[\text{city}] \wedge t_1[\text{city}] = \text{EDI}\big)\Big)\end{aligned}$$

Conditional Functional Dependencies (CFDs)

Example (Conditional Functional Dependency (CFD))

"In the UK, if the area code is 131, then the city must be Edinburgh (EDI)"

$$\forall t_1 \forall t_2 \Big(\big(\text{Address}(t_1) \wedge \text{Address}(t_2) \wedge$$
$$t_1[\text{CC}] = t_2[\text{CC}] \wedge t_1[\text{AC}] = t_2[\text{AC}] \wedge t_1[\text{CC}] = 44 \wedge t_1[\text{AC}] = 131\big)$$
$$\rightarrow \big(t_1[\text{city}] = t_2[\text{city}] \wedge t_1[\text{city}] = \text{EDI}\big)\Big)$$

Compact Notation

$\text{Address}[\text{CC} = 44, \text{AC} = 131 \rightarrow \text{city} = \text{EDI}]$

Conditional Functional Dependencies (CFDs)

Example (Conditional Functional Dependency (CFD))

"In the USA, if the area code is 908, then the city must be Murray Hill (MH)"

$$\forall t_1 \forall t_2 \Big(\big(\mathsf{Address}(t_1) \wedge \mathsf{Address}(t_2) \wedge t_1[\mathsf{CC}] = t_2[\mathsf{CC}] \wedge t_1[\mathsf{AC}] = t_2[\mathsf{AC}] \wedge t_1[\mathsf{CC}] = 01 \wedge t_1[\mathsf{AC}] = 908\big) \rightarrow \big(t_1[\mathsf{city}] = t_2[\mathsf{city}] \wedge t_1[\mathsf{city}] = \mathsf{MH}\big) \Big)$$

Conditional Functional Dependencies (CFDs)

Example (Conditional Functional Dependency (CFD))

"In the USA, if the area code is 908, then the city must be Murray Hill (MH)"

$$\forall t_1 \forall t_2 \Big(\big(\mathsf{Address}(t_1) \land \mathsf{Address}(t_2) \land \\ t_1[\mathsf{CC}] = t_2[\mathsf{CC}] \land t_1[\mathsf{AC}] = t_2[\mathsf{AC}] \land t_1[\mathsf{CC}] = 01 \land t_1[\mathsf{AC}] = 908\big) \\ \rightarrow \big(t_1[\mathsf{city}] = t_2[\mathsf{city}] \land t_1[\mathsf{city}] = \mathsf{MH}\big) \Big)$$

Compact Notation

$\mathsf{Address}[\mathsf{CC} = 01, \mathsf{AC} = 908 \rightarrow \mathsf{city} = \mathsf{MH}]$

Conditional Functional Dependencies (CFDs)

Example

Relation Address

CC	AC	phn	name	street	city	zip
44	131	1234567	Mike	Mayfield	NYC	EH4 8LE
44	131	3456789	Sarah	Crichton	NYC	EH4 8LE
01	908	3456789	Alex	Mtn Ave	NYC	07974

Conditional Functional Dependencies

- $\text{Address}[\text{CC} = 44, \text{zip} \rightarrow \text{street}]$
- $\text{Address}[\text{CC} = 44, \text{AC} = 131 \rightarrow \text{city} = \text{EDI}]$
- $\text{Address}[\text{CC} = 01, \text{AC} = 908 \rightarrow \text{city} = \text{MH}]$

Observations

- All tuples in the relation are dirty
- But it satisfies all the FDs from earlier!

Extending Inclusion Dependencies

Example (Inclusion Dependency)

Item[asin, title, price] $\subseteq$ Book[asin, title, price]

Database Instance

Relation Book

asin	isbn	title	price
a23	b32	Le Petit Prince	17,99€
a56	b65	Snow White	7,94€

Relation Item

asin	title	type	price
a23	Le Petit Prince	book	17,99€
a12	John Denver	CD	7,94€

Extending Inclusion Dependencies

Example (Inclusion Dependency)

$\text{Item}[\text{asin, title, price}] \subseteq \text{Book}[\text{asin, title, price}]$

Observations

- These instances do not satisfy the IND
  - There is no tuple for the item with asin a12 in the Book relation
  - This item is a CD, not a book!

Database Instance

Relation Book

asin	isbn	title	price
a23	b32	Le Petit Prince	17,99€
a56	b65	Snow White	7,94€

Relation Item

asin	title	type	price
a23	Le Petit Prince	book	17,99€
a12	John Denver	CD	7,94€

Extending Inclusion Dependencies

Example (Inclusion Dependency)

Item[asin, title, price] $\subseteq$ Book[asin, title, price]

Observations

- These instances do not satisfy the IND
  - There is no tuple for the item with asin a12 in the Book relation
  - This item is a CD, not a book!

Database Instance

Relation Book

asin	isbn	title	price
a23	b32	Le Petit Prince	17,99€
a56	b65	Snow White	7,94€

Relation Item

asin	title	type	price
a23	Le Petit Prince	book	17,99€
a12	John Denver	CD	7,94€

Semantic Property

"The IND only makes sense for tuples corresponding to books"

Reminder – Inclusion Dependencies as First-Order Sentences

INDs as First-Order Sentences

An inclusion dependency

$$R[A_1, \ldots, A_m] \subseteq S[B_1, \ldots, B_m]$$

can be written as

$$\forall t \Big(R(t) \rightarrow \exists s \big(S(s) \wedge \bigwedge_{i \in [1,m]} t[A_i] = s[B_i] \big) \Big)$$

Reminder – Inclusion Dependencies as First-Order Sentences

INDs as First-Order Sentences

An inclusion dependency

$$R[A_1, \ldots, A_m] \subseteq S[B_1, \ldots, B_m]$$

can be written as

$$\forall t \Big(R(t) \to \exists s \big(S(s) \wedge \bigwedge_{i \in [1,m]} t[A_i] = s[B_i] \big) \Big)$$

Idea

To express the previous semantic property in a similar formalism we add equality with constants

Conditional Inclusion Dependencies

Example (Conditional Inclusion Dependencies)

"The IND $\mathrm{Item}[\mathrm{asin},\ \mathrm{title},\ \mathrm{price}] \subseteq \mathrm{Book}[\mathrm{asin},\ \mathrm{title},\ \mathrm{price}]$ only holds for books."

$$\forall t\Big(\big(\mathrm{Item}(t) \wedge t[\mathrm{type}] = \mathrm{book}\big) \\ \rightarrow \exists s\big(\mathrm{book}(s) \wedge t[\mathrm{asin}] = s[\mathrm{asin}] \wedge t[\mathrm{title}] = s[\mathrm{title}] \wedge t[\mathrm{price}] = s[\mathrm{price}]\big)\Big)$$

Conditional Inclusion Dependencies

Example (Conditional Inclusion Dependencies)

"The IND Item[asin, title, price] $\subseteq$ Book[asin, title, price] only holds for books."

$$\forall t\Big(\big(\mathrm{Item}(t) \land t[\mathrm{type}] = \mathrm{book}\big) \\ \rightarrow \exists s\big(\mathrm{book}(s) \land t[\mathrm{asin}] = s[\mathrm{asin}] \land t[\mathrm{title}] = s[\mathrm{title}] \land t[\mathrm{price}] = s[\mathrm{price}]\big)\Big)$$

Compact Notation

$\mathrm{Item}[\mathrm{asin}, \mathrm{title}, \mathrm{price}, \mathrm{type} = \mathrm{book}] \subseteq \mathrm{Book}[\mathrm{asin}, \mathrm{title}, \mathrm{price}]$

Conditional Inclusion Dependencies

Example (Conditional Inclusion Dependencies)

"The IND Item[asin, title, price] $\subseteq$ Book[asin, title, price] only holds for books."

$$\forall t\Big(\big(\mathrm{Item}(t) \wedge t[\mathrm{type}] = \mathrm{book}\big) \\ \rightarrow \exists s\big(\mathrm{book}(s) \wedge t[\mathrm{asin}] = s[\mathrm{asin}] \wedge t[\mathrm{title}] = s[\mathrm{title}] \wedge t[\mathrm{price}] = s[\mathrm{price}]\big)\Big)$$

Compact Notation

$\mathrm{Item}[\mathrm{asin}, \mathrm{title}, \mathrm{price}, \mathrm{type} = \mathrm{book}] \subseteq \mathrm{Book}[\mathrm{asin}, \mathrm{title}, \mathrm{price}]$

Observation

Similarly to CFDs, we add **conditions** to inclusion dependencies

Capturing Inconsistencies Across Relations

Example (Conditional Inclusion Dependency)

$\text{Item}[\text{asin}, \text{title}, \text{price}, \text{type} = \text{book}] \subseteq \text{Book}[\text{asin}, \text{title}, \text{price}]$

Database Instance

Relation Book

asin	isbn	title	price
a23	b32	Le Petit Prince	17,99€
a56	b65	Snow White	7,94€

Relation Item

asin	title	type	price
a23	Le Petit Prince	book	17,99€
a12	John Denver	CD	7,94€

Capturing Inconsistencies Across Relations

Example (Conditional Inclusion Dependency)

$\text{Item}[\text{asin}, \text{title}, \text{price}, \text{type} = \text{book}] \subseteq \text{Book}[\text{asin}, \text{title}, \text{price}]$

Observations

- The database instance satisfies the conditional inclusion dependency
- but not the original IND
- Conditional inclusion dependencies are better suited for improving the quality of data

Database Instance

Relation Book

asin	isbn	title	price
a23	b32	Le Petit Prince	17,99€
a56	b65	Snow White	7,94€

Relation Item

asin	title	type	price
a23	Le Petit Prince	book	17,99€
a12	John Denver	CD	7,94€

Matching Dependencies for Record Matching

Record Matching/Object Identification

- Identification of tuples from one or more relations that refer to the same real-world object
- Applied to improve data quality and for data integration

Record Matching

Record Matching/Object Identification

- Identification of tuples from one or more relations that refer to the same real-world object
- Applied to improve data quality and for data integration

Example (Credit Card Fraud Detection)

Relation CardHolder

FN	LN	address	tel	DoB
Mark	Smith	10 Oak St, EDI, EH8 9LE	3256777	10/12/97

Relation Transaction

FN	LN	post	phn	when	amount
M.	Smith	10 Oak St, EDI, EH8 9LE	`null`	1pm 7/7/09	£3500
⋮	⋮	⋮	⋮	⋮	⋮
Max	Smith	PO Box 25, EDI	3256777	2pm 7/7/09	£6300

Record Matching

Example (Credit Card Fraud Detection)

Relation CardHolder

FN	LN	address	tel	DoB
Mark	Smith	10 Oak St, EDI, EH8 9LE	3256777	10/12/97

Relation Transaction

FN	LN	post	phn	when	amount
M.	Smith	10 Oak St, EDI, EH8 9LE	`null`	1pm 7/7/09	£3500
⋮	⋮	⋮	⋮	⋮	⋮
Max	Smith	PO Box 25, EDI	3256777	2pm 7/7/09	£6300

Statistics

World-wide losses in fraud in 2006: $4.84 billion (source: SAS)

Record Matching

Example (Credit Card Fraud Detection)

Relation CardHolder

FN	LN	address	tel	DoB
Mark	Smith	10 Oak St, EDI, EH8 9LE	3256777	10/12/97

Relation Transaction

FN	LN	post	phn	when	amount
M.	Smith	10 Oak St, EDI, EH8 9LE	`null`	1pm 7/7/09	£3500
⋮	⋮	⋮	⋮	⋮	⋮
Max	Smith	PO Box 25, EDI	3256777	2pm 7/7/09	£6300

A Non-Trivial Problem

- Real-life data is often dirty: errors are in the data sources
- Data is often represented differently in different sources
- Pairwise comparison of attributes **via equality** only is **not** sufficient

Matching Dependencies (MDs)[2]

Example (Matching Dependencies (MDs))

> *"If two entities (tuples) agree on their last name and address and if their first names are similar, then the two tuples should be identified on related attributes"*

---

[2]Fan, "Dependencies revisited for improving data quality", *Proceedings of the Twenty-Seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS*,. 2008
Bertossi et al., "Data Cleaning and Query Answering with Matching Dependencies and Matching Functions", *Theory Comput. Syst.*, 2013

Matching Dependencies (MDs)[2]

Example (Matching Dependencies (MDs))

"If two entities (tuples) agree on their last name and address and if their first names are similar, then the two tuples should be identified on related attributes"

$$\forall s \forall t \Big(\big(\mathrm{CardHolder}(s) \wedge \mathrm{Transaction}(t) \\ \wedge\, s[\mathrm{LN}] = t[\mathrm{LN}] \wedge s[\mathrm{address}] = t[\mathrm{post}] \wedge s[\mathrm{FN}] \asymp t[\mathrm{FN}]\big) \rightarrow s[X] = t[Y] \Big)$$

- $\asymp$ is a similarity operator
- X and Y are compatible attributes of CardHolder and Transaction, respectively

---

[2]Fan, "Dependencies revisited for improving data quality", *Proceedings of the Twenty-Seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS*,. 2008
Bertossi et al., "Data Cleaning and Query Answering with Matching Dependencies and Matching Functions", *Theory Comput. Syst.*, 2013

Matching Dependencies (MDs)

Example (Matching Dependencies (MDs))

"If two entities (tuples) agree on their last name and address and if their first names are similar, then the two tuples should be identified on related attributes"

$$\forall s \forall t \Big(\big(\mathrm{CardHolder}(s) \wedge \mathrm{Transaction}(t) \\ \wedge\, s[\mathrm{LN}] = t[\mathrm{LN}] \wedge s[\mathrm{address}] = t[\mathrm{post}] \wedge s[\mathrm{FN}] \asymp t[\mathrm{FN}]\big) \rightarrow s[X] = t[Y] \Big)$$

- $\asymp$ is a similarity operator
- X and Y are compatible attributes of CardHolder and Transaction, respectively

MDs vs. FDs

A matching dependency is similar to an FD, but

- equalities can be relaxed to similarities; and
- it can relate multiple relations

Example (Reasoning)

Given the two MDs

$$\forall s \forall t \Big(\big(\mathrm{CardHolder}(s) \wedge \mathrm{Transaction}(t) \\ \wedge s[\mathrm{LN}] = t[\mathrm{LN}] \wedge s[\mathrm{address}] = t[\mathrm{post}] \wedge s[\mathrm{FN}] \asymp t[\mathrm{FN}]\big) \rightarrow s[X] = t[Y] \Big)$$

and

$$\forall s \forall t \Big(\big(\mathrm{CardHolder}(s) \wedge \mathrm{Transaction}(t) \wedge s[\mathrm{tel}] = t[\mathrm{phone}]\big) \rightarrow s[\mathrm{address}] = t[\mathrm{post}] \Big)$$

Example (Reasoning)

Given the two MDs

$$\forall s \forall t \Big(\big(\mathrm{CardHolder}(s) \wedge \mathrm{Transaction}(t) \\ \wedge\ s[\mathrm{LN}] = t[\mathrm{LN}] \wedge s[\mathrm{address}] = t[\mathrm{post}] \wedge s[\mathrm{FN}] \asymp t[\mathrm{FN}]\big) \to s[X] = t[Y]\Big)$$

and

$$\forall s \forall t \Big(\big(\mathrm{CardHolder}(s) \wedge \mathrm{Transaction}(t) \wedge s[\mathrm{tel}] = t[\mathrm{phone}]\big) \to s[\mathrm{address}] = t[\mathrm{post}]\Big)$$

we may infer a third MD

$$\forall s, t \Big(\big(\mathrm{CardHolder}(s) \wedge \mathrm{Transaction}(t) \\ \wedge\ s[\mathrm{LN}] = t[\mathrm{LN}] \wedge s[\mathrm{tel}] = t[\mathrm{phone}] \wedge s[\mathrm{FN}] \asymp t[\mathrm{FN}]\big) \to s[X] = t[Y]\Big)$$

How are MDs used for Matching?

Dynamic Semantics

- Matching tuples are obtained from an instance that does not satisfy the MDs
- Matching keys: A minimal set of attributes that allow for matching two tuples

How are MDs used for Matching?

Dynamic Semantics

- Matching tuples are obtained from an instance that does not satisfy the MDs
- Matching keys: A minimal set of attributes that allow for matching two tuples

Example (Credit Card Fraud Detection)

Relation CardHolder

FN	LN	address	tel	DoB
Mark	Smith	10 Oak St, EDI, EH8 9LE	3256777	10/12/97

Relation Transaction

FN	LN	post	phn	when	amount
M.	Smith	10 Oak St, EDI, EH8 9LE	`null`	1pm 7/7/09	£3500
⋮	⋮	⋮	⋮	⋮	⋮
Max	Smith	PO Box 25, EDI	3256777	2pm 7/7/09	£6300

How are MDs used for Matching?

Dynamic Semantics

- Matching tuples are obtained from an instance that does not satisfy the MDs
- Matching keys: A minimal set of attributes that allow for matching two tuples

Example (Credit Card Fraud Detection)

Relation CardHolder

FN	LN	address	tel	DoB
Mark	Smith	10 Oak St, EDI, EH8 9LE	3256777	10/12/97

Relation Transaction

FN	LN	post	phn	when	amount
M.	Smith	10 Oak St, EDI, EH8 9LE	3256777	1pm 7/7/09	£3500
⋮	⋮	⋮	⋮	⋮	⋮
Max	Smith	PO Box 25, EDI	3256777	2pm 7/7/09	£6300

How are MDs used for Matching?

Dynamic Semantics

- Matching tuples are obtained from an instance that does not satisfy the MDs
- Matching keys: A minimal set of attributes that allow for matching two tuples

Example (Credit Card Fraud Detection)

Relation CardHolder

FN	LN	address	tel	DoB
Mark	Smith	10 Oak St, EDI, EH8 9LE	3256777	10/12/97

Relation Transaction

FN	LN	post	phn	when	amount
M.	Smith	10 Oak St, EDI, EH8 9LE	3256777	1pm 7/7/09	£3500
⋮	⋮	⋮	⋮	⋮	⋮
Max	Smith	PO Box 25, EDI	3256777	2pm 7/7/09	£6300

How are MDs used for Matching?

Dynamic Semantics

- Matching tuples are obtained from an instance that does not satisfy the MDs
- Matching keys: A minimal set of attributes that allow for matching two tuples

Example (Credit Card Fraud Detection)

Relation CardHolder

FN	LN	address	tel	DoB
Mark	Smith	10 Oak St, EDI, EH8 9LE	3256777	10/12/97

Relation Transaction

FN	LN	post	phn	when	amount
M.	Smith	10 Oak St, EDI, EH8 9LE	3256777	1pm 7/7/09	£3500
⋮	⋮	⋮	⋮	⋮	⋮
Mark	Smith	10 Oak St, EDI, EH8 9LE	3256777	2pm 7/7/09	£6300

Key Features of Matching Dependencies

Summary

- MDs have a dynamic semantics: they actually change tuples by means of identification
- MDs allow for automated reasoning: to infer new MDs and identify matching keys
- MDs have a formalism that is similar to that of CFDs: integration of consistency and matching

Key Features of Matching Dependencies

Summary

- MDs have a dynamic semantics: they actually change tuples by means of identification
- MDs allow for automated reasoning: to infer new MDs and identify matching keys
- MDs have a formalism that is similar to that of CFDs: integration of consistency and matching

Current State

Both theoretical and practical aspects of MDs have been investigated, but their interaction with other data quality aspects needs to be explored further

Other Kinds of Dependencies

Currency Dependencies (CDs)[2]

Example (Currency Dependency (CD))

"Divorce comes after marriage"

$$\forall t_1 \forall t_2 \Big(\big(\text{Resident}(t_1) \wedge \text{Resident}(t_2) \\ \wedge\ t_1[\text{eid}] = t_2[\text{eid}] \wedge t_2[\text{status}] = \text{divorced} \wedge t_1[\text{status}] = \text{married}\big) \rightarrow t_1 \prec_{\mathit{status}} t_2 \Big)$$

[2]Fan et al., "Determining the Currency of Data", *ACM Trans. Database Syst.*, 2012

Currency Dependencies (CDs)[2]

Example (Currency Dependency (CD))

"Divorce comes after marriage"

$$\forall t_1 \forall t_2 \Big(\big(\text{Resident}(t_1) \wedge \text{Resident}(t_2) \\ \wedge\ t_1[\text{eid}] = t_2[\text{eid}] \wedge t_2[\text{status}] = \text{divorced} \wedge t_1[\text{status}] = \text{married}\big) \rightarrow t_1 \prec_{status} t_2 \Big)$$

Currency Dependencies (CDs)

- CDs extend FDs by allowing a temporal partial order $\prec_A$ on each attribute A
  - $t_1 \prec_A t_2$ holds if $t_2[A]$ is more recent than $t_1[A]$
- Semantic properties of the data are used to infer temporal orderings
- For ensuring timeliness of data

[2] Fan et al., "Determining the Currency of Data", *ACM Trans. Database Syst.*, 2012

Editing Rules (eRs)

Example (Editing Rule (eR))

"If we know that the zip code of a tuple is correct, and if a user can provide the correct area code, street, and city for that zip code, then take the values from the user"

$$\forall t_1 \forall t_2 \Big(\big(\text{Address}(t_1) \land \text{Address}_u(t_2) \land t_1[\text{zip}] = t_2[\text{zip}]\big) \\ \rightarrow \big(t_1[\text{AC}] = t_2[\text{AC}] \land t_1[\text{street}] = t_2[\text{street}] \land t_1[\text{city}] = t_2[\text{city}]\big)\Big)$$

Editing Rules (eRs)

Example (Editing Rule (eR))

"If we know that the zip code of a tuple is correct, and if a user can provide the correct area code, street, and city for that zip code, then take the values from the user"

$$\forall t_1 \forall t_2 \Big(\big(\text{Address}(t_1) \wedge \text{Address}_u(t_2) \wedge t_1[\text{zip}] = t_2[\text{zip}]\big) \\ \rightarrow \big(t_1[\text{AC}] = t_2[\text{AC}] \wedge t_1[\text{street}] = t_2[\text{street}] \wedge t_1[\text{city}] = t_2[\text{city}]\big)\Big)$$

Editing Rules (eRs)

- eRs provide dynamic semantics on top of CFDs; and
- incorporate user interaction
  - by means of special relation symbols R_u

Other Kinds of Data Quality Dependencies

Sequential Dependencies

Can enforce limited variation in streaming data

Metric Dependencies

Can restrict the distance between values
in the consequence of a dependency according to some metric

And many more

...

Key Algorithmic Challenges

Discovering Data Quality Dependencies

Where do dependencies come from?

- Manual design (expensive and time consuming)
- Business rules (not expressive enough)

Discovering Data Quality Dependencies

Where do dependencies come from?

- Manual design (expensive and time consuming)
- Business rules (not expressive enough)

The Discovery Problem

Given a sample of the data, find a cover of data quality dependencies that hold on the sample

Discovering Data Quality Dependencies

Where do dependencies come from?

- Manual design (expensive and time consuming)
- Business rules (not expressive enough)

The Discovery Problem

Given a sample of the data, find a cover of data quality dependencies that hold on the sample

- Several effective algorithms for discovering conditional and matching dependencies are already in place
- Goal: Automatic discovery of data quality dependencies

The Implication Problem

Input: a set of data quality dependencies dependencies

Output: all dependencies that are implied by the given set

Reasoning over Data Quality Dependencies

The Implication Problem

Input: a set of data quality dependencies dependencies

Output: all dependencies that are implied by the given set

Optimizations

Reasoning allows for

- reducing the number of dependencies
- inferring new knowledge

Reasoning over Data Quality Dependencies

The Implication Problem

Input: a set of data quality dependencies dependencies
Output: all dependencies that are implied by the given set

Optimizations

Reasoning allows for

- reducing the number of dependencies
- inferring new knowledge

Goal

Automated methods for reasoning about data quality dependencies

Error Detection: SQL-Based Techniques

The Error Detection Problem

Input: a set data quality dependencies and a database

Output: tuples that violate the dependencies

Error Detection: SQL-Based Techniques

The Error Detection Problem

Input: a set data quality dependencies and a database

Output: tuples that violate the dependencies

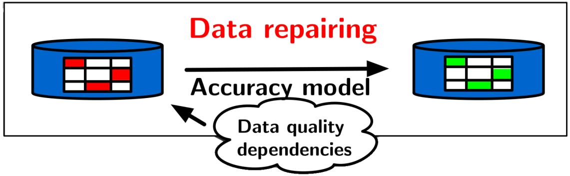
- Efficient methods are devised in both centralized and distributed setting for CFDs and CINDs
- Goal: Automatically check whether the data is dirty or clean

Error Repairing: Fixing the Errors Discovered

The Error Repairing Problem

Input: a set data quality dependencies and a database D

Output: a database D', called repair of D, satisfying all dependencies, and such that quality(D, D') is maximal

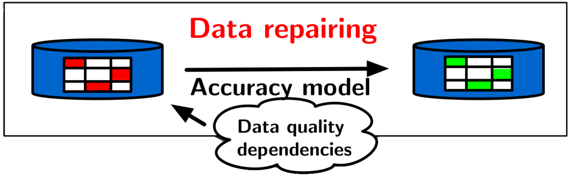


Error Repairing: Fixing the Errors Discovered

The Error Repairing Problem

Input: a set data quality dependencies and a database D

Output: a database D', called repair of D, satisfying all dependencies, and such that quality(D, D') is maximal



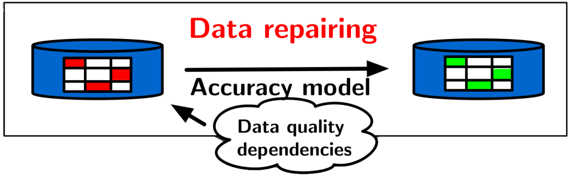
- Quality metrics take into account
  - weights associated with attributes
  - distances between the original and updated values

Error Repairing: Fixing the Errors Discovered

The Error Repairing Problem

Input: a set data quality dependencies and a database D

Output: a database D', called repair of D, satisfying all dependencies, and such that quality(D, D') is maximal



- Quality metrics take into account
  - weights associated with attributes
  - distances between the original and updated values
- Performance guarantees
  - The problem is NP-complete, even for a fixed set of FDs; hence use of heuristics
  - accuracy above a predefined precision with a high confidence
  - user inspection and feedback, proportional to the time allocated

How to Fix Errors?

Observations

- Dependencies indicate possible repairs (by chasing the database with them), but this may lead to many and often inaccurate or simply incorrect repairs
- Certain fixes: 100% correct. The need for this is evident when repairing critical data
  - Every update guarantees to fix an error
  - The repairing process does not introduce new errors

Editing Rules and Certain Attributes

Editing rules are data quality dependencies that

- tell which values to select in repairs (by leveraging reliable reference data)
- only chase when the premise of the dependency contains certified attributes only

How to Fix Errors?

Interaction

Repairing and record matching should be intertwined

- Repairing can help matching
- Matching can help repairing

In Reality

In practice, customers are hesitant to see their data automatically cleaned

- Suggested repairs are welcome
- Certain fixes that use reference data only is acceptable

Summary

Data quality

The No.1 problem for data management

- Real life data is dirty, dirty data is costly
- The quest for a principled approach
- Many application scenarios

Remaining Challenges

- Effective algorithms for certain fixes (minimum user interaction)
- Data completeness
- Data currency
- Data accuracy
- Putting it all together: Interaction between central issues of data quality

Summary

Data quality

The No.1 problem for data management

- Real life data is dirty, dirty data is costly
- The quest for a principled approach
- Many application scen

Take away message

Data quality: a rich source of problems and challenges

Remaining Ch

- Effective al… for certain fixes (minimum user interaction)
- Data completeness
- Data currency
- Data accuracy
- Putting it all together: Interaction between central issues of data quality

References

Bertossi, Leopoldo E., Solmaz Kolahi, and Laks V. S. Lakshmanan. "Data Cleaning and Query Answering with Matching Dependencies and Matching Functions". In: *Theory Comput. Syst.* 52.3 (2013), pp. 441–482. DOI: 10.1007/S00224-012-9402-7.

Fan, Wenfei. "Dependencies revisited for improving data quality". In: *Proceedings of the Twenty-Seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS*. Ed. by Maurizio Lenzerini and Domenico Lembo. ACM, 2008, pp. 159–170. DOI: 10.1145/1376916.1376940.

Fan, Wenfei, Floris Geerts, and Jef Wijsen. "Determining the Currency of Data". In: *ACM Trans. Database Syst.* 37.4 (2012), 25:1–25:46. DOI: 10.1145/2389241.2389244.