CPT-S 415

Big Data

Yinghui Wu

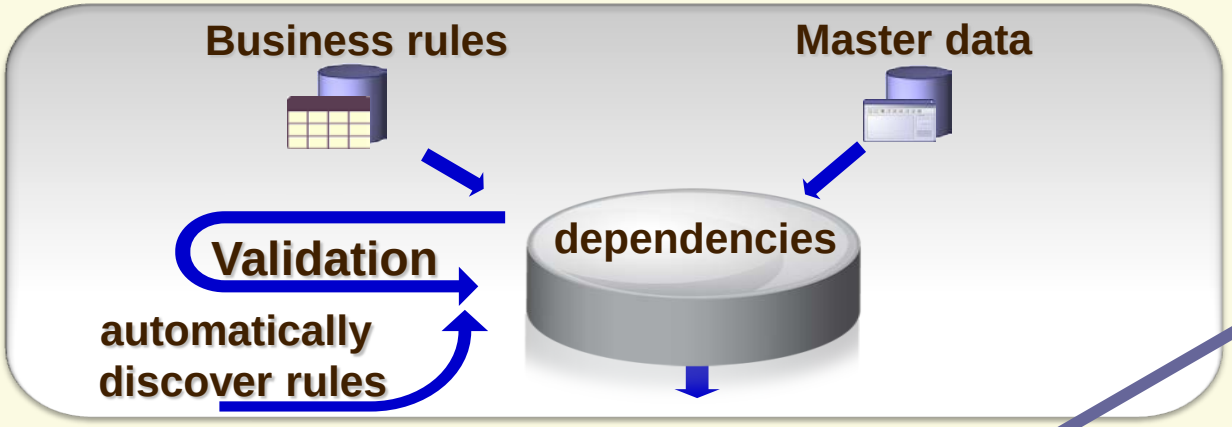
EME B45

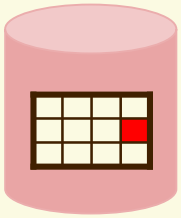
CPT-S 415
Big Data

Data cleaning

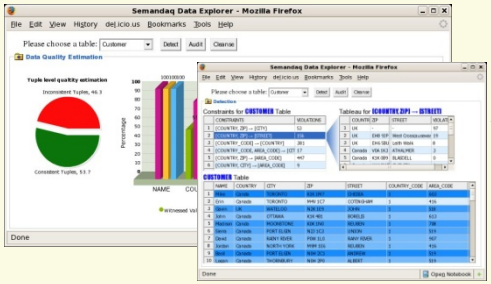
- ✓ Data cleaning: An overview
- ✓ Error detection
- ✓ Data repairing
- ✓ Record matching and its interaction with data repairing
- ✓ Certain fixes

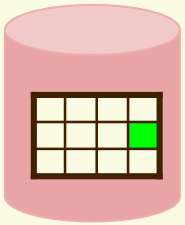
A platform for improving data quality





Dirty data





Clean Data

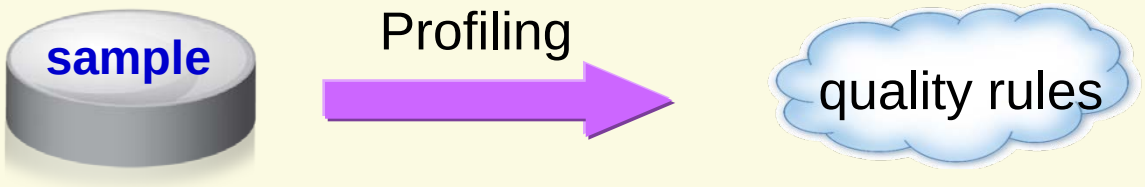
- profiling
- validating
- error detecting
- data repairing
- record matching
- certain fixes

- Standardization
- Auditing
- Enrichment
- Monitoring
- Data explorer

A practical data cleaning system

Profiling: Discovering conditional dependencies

- Input: sample data D
- Output: a cover of conditional dependencies that hold on D



Several effective algorithms for discovering conditional dependencies and matching dependencies are already in place

Question: automatic discovery of conditional inclusion dependencies?

Automatic discovery of data quality rules

Assessing the quality of conditional dependencies

- Input: a set of conditional functional dependencies
- Output: a maximum satisfiable subset of dependencies

are the dependencies discovered "dirty" themselves?

Complexity: the MAXSC probl

Theorem: there is an ε-approximation algorithm for MAXSC

- ✓ there exist constant ε such that for the subset $\sum_m$ found by the algorithm has a bound: $card(\sum_m) > \varepsilon\ card(OPT(\sum))$

Approximation algorithms

- ✓ Efficient: low polynomial time
- ✓ Performance guarantee

Automated methods for reasoning about data quality rules

Profiling: Discovering conditional dependencies

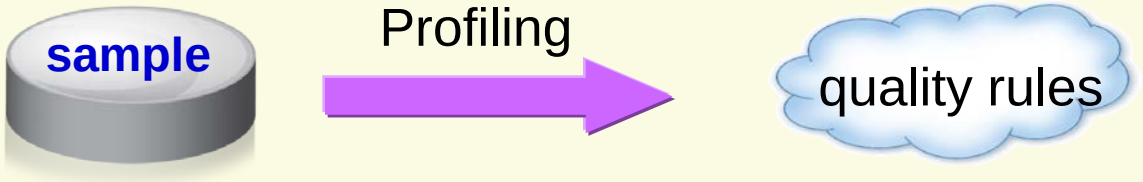
Where do dependencies (data quality rules) come from?

- ✓ Manual design: domain knowledge analysis
- ✓ Business rules
- ✓ Discovery

Expensive

Inadequate

- Input: sample data D
- Output: a cover of conditional dependencies that hold on D



Several effective algorithms for discovering conditional dependencies, developed by researchers in the UK, Canada and the US (e.g., AT&T)

Automatic discovery of data quality rules

Error detection

Detecting violations of CFDs

$\varphi 1$ [CC=44, zip] →[street]
$\varphi 2$ [CC=44, AC=131] →[city=Edi]

tid	name	title	CC	AC	phn	str	city	zip
t1	Mike	MTS	44	131	1234567	Mayfield	NYC	EH4 8LE
t2	Rick	DMTS	44	131	3456789	Chrichton	NYC	EH4 8LE
t3	Phil	DMTS	44	131	2909229	Chrichton	Edi	EH4 8LE

Input: A set ∑ of CFDs, and a database D

Output: All tuples in D that violate at least one CFD in ∑

Automatically check whether the data is dirty or clean

Detecting CFD violations

- ✓ Input: a set $\sum$ of CFDs and a database DB
- ✓ Output: the set of tuples in DB that violate at least one CFD in $\sum$

Approach: automatically generate SQL queries to find violations

Complication 1: consider (R: X $\rightarrow$ Y, Tp), the pattern tableau may be large (recall that each tuple in Tp is in fact a constraint)

Goal: the size of the SQL queries is independent of Tp

Trick: treat Tp as a data table

CINDs can be checked along the same lines

Single CFD: step 1

A pair of SQL queries, treating Tp as a data table

- Single-tuple violation (pattern matching)
- Multi-tuple violations (traditional FDs)

(cust(country, area-code, phone → street, city, zip), Tp)

✓ Single-tuple violation: Qc

Testing pattern tuples

```
select   *
from     R  t,  Tp  tp
where    t[country] ≈ tp[country] AND t[area-code] ≈ tp[area-code]
           AND t[phone] ≈ tp[phone]
         (t[street] <> tp[street] OR t[city] <> tp[city] OR t[zip] <> tp[zip]))
```

- <>: not matching;
- t[A1] ≈ tp[A1]: (t[A1] = tp[A1] OR tp[A1] = _)

Single CFD: step 2

(cust(country, area-code, phone → street, city, zip), Tp)

- ✓ Multi-tuple violations (the semantics of traditional FDs): Qv

```
select   distinct  t.country, t.area-code, t.phone
from      R  t,  Tp  tp
where    t[country] ≈ tp[country] AND t[area-code] ≈ tp[area-code]
          AND t[phone] ≈ tp[phone]
group by   t.country, t.area-code, t.phone
having  count(distinct street, city, zip) > 1
```

The semantics of FDs

Tp is treated as a data table

Multiple CFDs

Complication 2: if the set $\sum$ has n CFDs, do we use 2n SQL queries, and thus 2n passes of the database DB?

Goal:

- ✓ 2 SQL queries no matter how many CFDs are in $\sum$
- ✓ the size of the SQL queries is independent of Tp

Trick: merge multiple CFDs into one

- ✓ Given (R: X1 $\rightarrow$ Y1, Tp1), (R: X2 $\rightarrow$ Y2, Tp2)
- ✓ Create a single pattern table: Tm = X1 $\cup$ X2 $\cup$ Y1 $\cup$ Y2,
- ✓ Introduce @, a don't-care variable, to populate attributes of pattern tuples in X1 – X2, etc (tp[A] = @)
- ✓ Modify the pair of SQL queries by using Tm

Handling multiple CFDs

CFD_1: (*area*→*state*, T_1)

area	state
_	_
212	NY

CFD_2: (*zip*→*state*, T_2)

zip	state
07974	NJ
90291	CA
01202	_

CFD_3: (*area,zip*→*state*, T_3)

area	zip	state
480	95120	CA
310	90995	CA

CFD_M:(*area,zip*→*state*, T_M)

	area	zip	state
CFD_2:	@	07974	NJ
CFD_2:	@	90291	CA
CFD_2:	@	01202	_
CFD_1:	_	@	_
CFD_1:	212	@	NY
CFD_3:	480	95120	CA
CFD_3:	310	90995	CA

Qc: select * **from** R t, T_M t_p **where**
t[*area*] ≍ t_p[*area*] **AND** t[*zip*] ≍ t_p[*zip*] **AND** t[*state*] ◇ t_p[*state*]

Qv: select distinct *area*, *zip* **from** Macro
group by *area*, *zip* **having count**(**distinct** *state*) > 1

Don't care

Macro:

select (**case** t_p[*area*] ***when*** "@" **then** "@" **else** t[*area*] **end**) as *area* . . .
from R t, T_M t_p
where t[*area*] ≍ t_p[*area*] **AND** t[*zip*] ≍ t_p[*zip*] **AND** t_p[state] =_

Detecting errors in horizontally partitioned data

$\varphi 1$ [CC=44, zip] →[street]
$\varphi 2$ [CC=44, AC=131] →[city=Edi]

distributed data

tid	name	title	CC	AC	phn	str	city	zip
t1	Mike	MTS	44	131	1234567	Mayfield	NYC	EH4 8LE

Horizontal partition

Partitioned by title

tid	name	title	CC	AC	phn	str	city	zip
t2	Rick	DMTS	44	131	3456789	Chrichton	NYC	EH4 8LE
t3	Phil	DMTS	44	131	2909229	Chrichton	Edi	EH4 8LE

To find violations of $\varphi 1$, it is necessary to ship data (part of t1 or t2) from one site to another

Detecting errors in vertically partitioned data

$\varphi 1$ [CC=44, zip] →[street]
$\varphi 2$ [CC=44, AC=131] →[city=Edi]

tid	name	title	CC	AC	phn	str	city	zip
t1	Mike	MTS	44	131	1234567	Mayfield	NYC	EH4 8LE
t2	Rick	DMTS	44	131	3456789	Chrichton	NYC	EH4 8LE
t3	Phil	DMTS	44	131	2909229	Chrichton	Edi	EH4 8LE

vertical partition

tid	name	title	CC	AC	phn
t1	[illegible]	[illegible]	44	131	1234567
t2	Rick	[illegible]	44	131	3456789
t3	Phil	DMTS	44	131	2909229

phone

tid	str	city	zip
t1	[illegible]vfield	NYC	EH4 8LE
t2	[illegible]	NYC	EH4 8LE
t3	Chr[illegible]chton	Edi	EH4 8LE

address

To find violations of $\varphi 2$, it is necessary to ship data

Error detection in distributed data

The error detection problem for CFDs is NP-complete for

- ✓ horizontally partitioned data, and
- ✓ vertically partitioned data,

when either minimum data shipment or minimum response time is concerned.

In contrast, error detection in centralized data is trivial

Heuristic (approximation) algorithms

Read: ***Detecting Inconsistencies in Distributed Data***

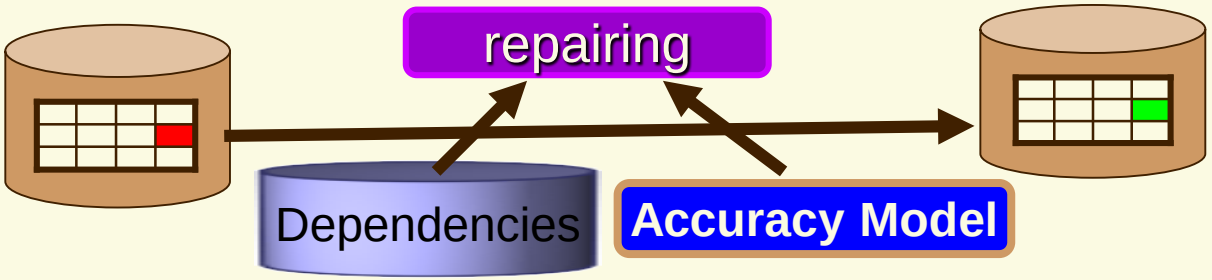
Error detection in distributed data is far more intriguing than its centralized counterpart

Data Repairing

Data repairing: Fixing the errors identified

- Input: a set Σ of conditional dependencies, and a database DB
- Output: a candidate repair DB' such that DB' satisfies Σ, and *cost*(*DB', DB*) is maximal

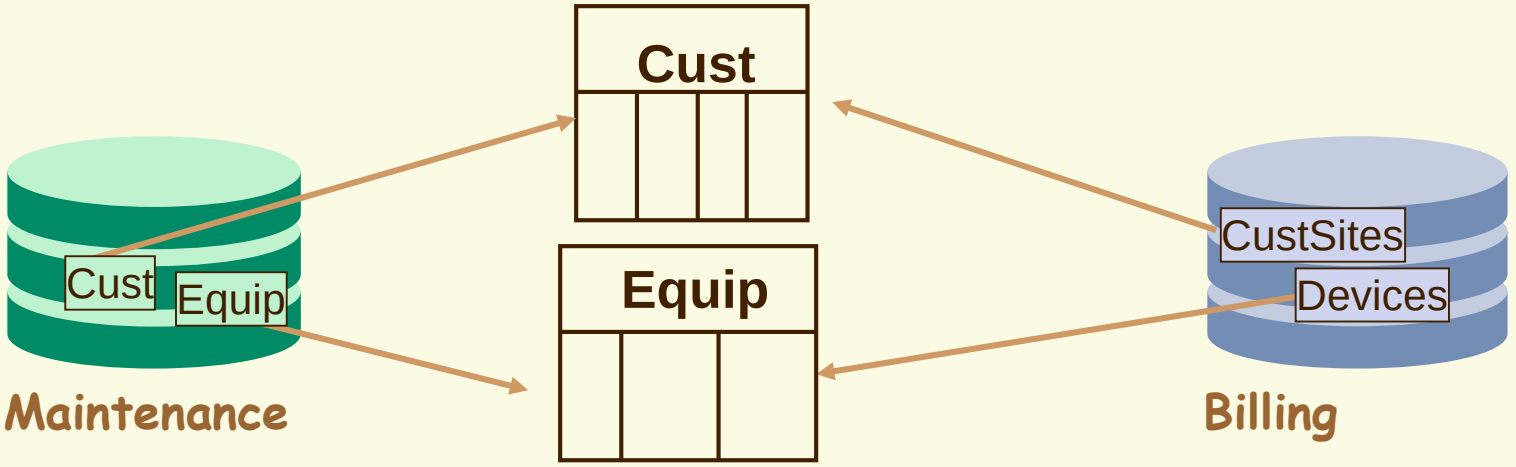
How to define?



The most challenging part of data cleaning

Example: networking service provider

- Assume the billing and maintenance departments have separate databases.
  - Internally consistent,
  - yet containing errors.
- Goal: reconcile and improve data quality of, for example, integrated customer and billing data.



Service provider example, continued.

cust

	phno	name	street	city	state	zip
t0	949-1212	Alice Smith	17 bridge	midville	az	05211
t1	555-8145	Bob Jones	5 valley rd	centre	ny	10012
t2	555-8195	Bob Jones	5 valley rd	centre	nj	10012
t3	212-6040	Carol Blake	9 mountain	davis	ca	07912
t4	949-1212	Ali Stith	27 bridge	midville	az	05211

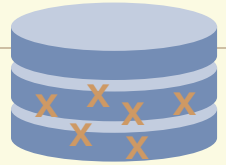
equip

	phno	serno	eqmfct	eqmodel	instdate
t5	949-1212	AC13006	AC	XE5000	Mar-02
t6	555-8145	L55001	LU	ze400	Jan-03
t7	555-8195	L55011	LU	ze400	Mar-03
t8	555-8195	AC22350	AC	XE5000	Feb-99
t9	949-2212	L32400	LU	ze300	Oct-01

Billing Dept.

Maintenance Dept.

Constraints and Violations (1)



- ✓ Consider inclusion and functional dependencies, (so each x is a violation of one or the other).
- ✓ A functional dependency (FD) says that the values of some fields determine the values of others.

t1 and t2 violate **cust[zip] -> cust[state]**

cust[phno] -> cust[name, street, city, state, zip]

cust

	phno	name	street	city	state	zip
t0	949-1212	Alice Smith	17 bridge	midville	az	05211
t1	555-8145	Bob Jones	5 valley rd	centre	ny	10012
t2	555-8195	Bob Jones	5 valley rd	centre	nj	10012
t3	212-6040	Carol Blake	9 mountain	davis	ca	07912
t4	949-1212	Ali Stith	27 bridge	midville	az	05211

Constraints and Violations (2)

An inclusion dependency (IND) says that the values of some fields should appear in some others.

t9 violates **equip[phno] ⊆ cust[phno]**

equip

	phno	serno	eqmfct	eqmodel	instdate
t9	949-2212	L32400	LU	ze300	Oct-01

cust

	phno	name	street	city	state	zip
t0	949-1212	Alice Smith	17 bridge	midville	az	05211
t1	555-8145	Bob Jones	5 valley rd	centre	ny	10012
t2	555-8195	Bob Jones	5 valley rd	centre	nj	10012
t3	212-6040	Carol Blake	9 mountain	davis	ca	07912
t4	949-1212	Ali Stith	27 bridge	midville	az	05211

Constraint Repair

- ✓ Find a “repair” to suggest
- ✓ A repair is a database which does not violate constraints
  - A good repair is also similar to the original database

Constraint 1: FD: R[A] -> R[B, C]

R:

A	B	C
a1	b1	c2
a2	b1	c1
a2	b1	c2

“Bad” repair:

A	B	C

"Good" repair:

A	B	C
a1	b1	c2
a2	b1	c1
a2	b1	c1

Problem Statement

Input: a relational database D, and a set C of integrity constraints (FDs, INDs)

Question: find a "good" repair D' of D

- ✓ repair: D' satisfies C
- ✓ "good": D' is "close" to the original data in D
  - changes are minimal: what metrics (referred to as cost) should we use?
  - changes: value modification, tuple insertion – to avoid loss of information (tuple deletion can be expressed via modification)

We want to compute D' efficiently, as suggested fix to the users

Repair Model

- For the duration of constraint repair, each input tuple is uniquely identified, t_1, t_2, …
- This value of attribute A of tuple t in the input database is D(t,A)
- The output of the algorithm is a repaired database, D', so D'(t,A) is the value of t.A in the current proposed repair.
- For example, D = D' below, except $D(t_3,C) <> D'(t_3,C)$.

D:

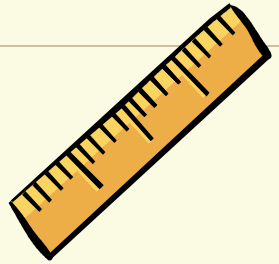
	A	B	C
t_1	a1	b1	c2
t_2	a2	b1	c1
t_3	a2	b1	c2

D':

	A	B	C
t_1	a1	b1	c2
t_2	a2	b1	c1
t_3	a2	b1	c1

Cost Model Components

Distance



Weight

Distant: "Smith" "Jones"

"a" "b"

Close: "949-2212" "949-1212"

"1.99" "2.0"

"GM" "General Motors"

Confidence placed by the user in the accuracy of a tuple or attribute.

dist(v,v') : [0,1)

weight(t)
weight(t.A)

Intuition: Tuple/Attribute Weights

- Simple model of data reliability
  - Example: Billing department may be more reliable about address information, weight = 2, but less about equipment, weight = 1
  - Example 2: Download table of zip codes from post-office web site, weight = 100
- In the absence of other information, all weights default to 1.

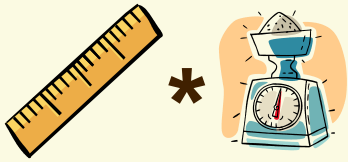
cust

	phno	name	street	city	state	zip	
t0	949-1212	Alice Smith	17 bridge	midville	az	05211	2
t1	555-8145	Bob Jones	5 valley rd	centre	ny	10012	2
t2	555-8195	Bob Jones	5 valley rd	centre	nj	10012	1
t3	212-6040	Carol Blake	9 mountain	davis	ca	07912	1
t4	949-1212	Ali Stith	27 bridge	midville	az	05211	1

Attribute-Level Cost Model

- If D(t.A) = v and D'(t.A) = v', then

  Cost(t.A) = dist(v, v') * weight(t.A)



  Cost(D'): the sum of Cost(t.A) for all changed tuples t and attributes A

- Example: (if we model delete as dist(x,null))

R:

A	B	C
a1	b1	c2
a2	b1	c1
a2	b1	c2

Repair: cost = 9

A	B	C
+1	+1	+1
+1	+1	+1
+1	+1	+1

Repair: cost = 1/2

A	B	C
a1	b1	c2
a2	b1	c1
a2	b1	c1 +1/2

Problem and complexity

Input: a relational database D, and a set C of integrity constraints

Question: find a repair D' of D such that cost(D') is minimal

Complexity: Finding an optimal repair is NP-complete in size of database

- ✓ Intractable even for a *small, constant* number of FDs alone.
- ✓ Intractable even for a *small, constant* number of INDs alone
- ✓ By contrast, in delete-only model, repair with either INDs or FDs alone is in PTime, while repair with both is CoNP hard

What should we do?

Heuristic approach to value-based repair

- In light of intractability, we turn to heuristic approaches.

  However, most have problems.

- Straightforward constraint-by-constraint repair algorithms fail to terminate (fixing individual constraints one by one)

  consider R1(A, B), R2(B, C), with
  - FD: R1[A] $\rightarrow$ R1[B]
  - IND: R2[B] $\subseteq$ R1[B]
  - D(R1): {(1, 2), (1, 3)}, D(R2) = {(2, 1), (3, 4)}

- Also, user must be involved since any single decision can be wrong.
- One approach: Equivalence-Class-Based Repair.

Equivalence Classes

cust

	phno	name	street	city	state	zip
t0	949-1212	Alice Smith	17 bridge	midville	az	05211
t1	555-8145	Bob Jones	5 valley rd	centre	ny	10012
t2	555-8195	Bob Jones	5 valley rd	centre	nj	10012
t3	212-6040	Carol Blake	9 mountain	davis	ca	07912
t4	949-1212	Ali Stith	27 bridge	midville	az	05211

eq1

- An equivalence class is a set of "cells" defined by a tuple t and an attribute A.
- An equivalence class eq has a target value targ(eq) drawn from eq -- to be assigned after all the equivalence classes are found
- For example, targ(EQ1) = "Alice Smith" or "Ali Stith"

Equivalence Classes Continued

- ✓ In the repair, give each member of the equivalence class the same target value: that is **for all** (t,A) in eq, D'(t,A) = targ(eq)
- ✓ Target value is chosen from the set of values associated with eq in the input: {D(t_i,Ai)}
- ✓ A given eq class has an easily computed cost w.r.t. a particular target value. For example, if weights are all 1, we might have,
  - Cost({"A","A","B"},"A") = dist("B","A") = 1
  - Cost({"A","A","B"},"B") = 2*dist("A","B") = 2

Separate

- ✓ The decision of which attribute values need to be equivalent
- ✓ The decision of exactly what value targ(eq) should be assigned

Resolving FD violations

- To resolve a tuple t violating FD R[**A**]->R[**B**], violations: compute the set of tuples V from R that match t on **A**, and union equivalence classes of the attributes in **B** among all tuples in V – changing RHS
  - Changing the left-hand side to an existing value may not resolve the violation
  - Changing left-hand side to a new value is arbitrary – loss of e.g., keys

Why?

FD: cust[phno] -> cust[name, street, city, state, zip]

	phno	name	street	city	state	zip
t0	949-1212	Alice Smith	17 bridge	midville	az	05211
t1	555-8145	Bob Jones	5 valley rd	centre	ny	10012
t2	555-8195	Bob Jones	5 valley rd	centre	nj	10012
t3	212-6040	Carol Blake	9 mountain	davis	ca	07912
t4	949-1212	Ali Stith	27 bridge	midville	az	05211

EQ1 EQ2 . . .

Resolving IND violations

- To resolve a tuple t violating IND R[**A**] ⊆ S[**B**], we pick a tuple s from S and union equivalence classes between t.A and s.B for attributes A in **A** and B in **B**.

$$equip[phno] \subseteq cust[phno]$$

EQ1

equip

	phno	serno	eqmfct	eqmodel	instdate
t9	949-2212	L32400	LU	ze300	Oct-01

cust

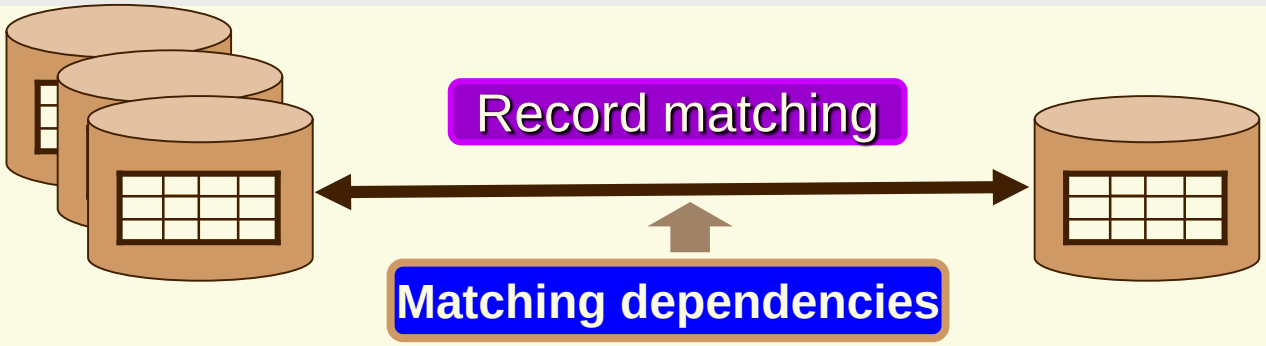
	phno	name	street	city	state	zip
t0	949-1212	Alice Smith	17 bridge	midville	az	05211
	...					
t3	212-6040	Carol Blake	9 mountain	davis	ca	07912
t4	949-1212	Ali Stith	27 bridge	midville	az	05211

How to repair data using CFDs? CINDs?

Interaction between Record matching & Data Repairing

Record matching

- Input: large, unreliable data sources
- Output: tuples that refer to the same real-world entity



Read: ***Interaction between Record Matching and Data Repairing***

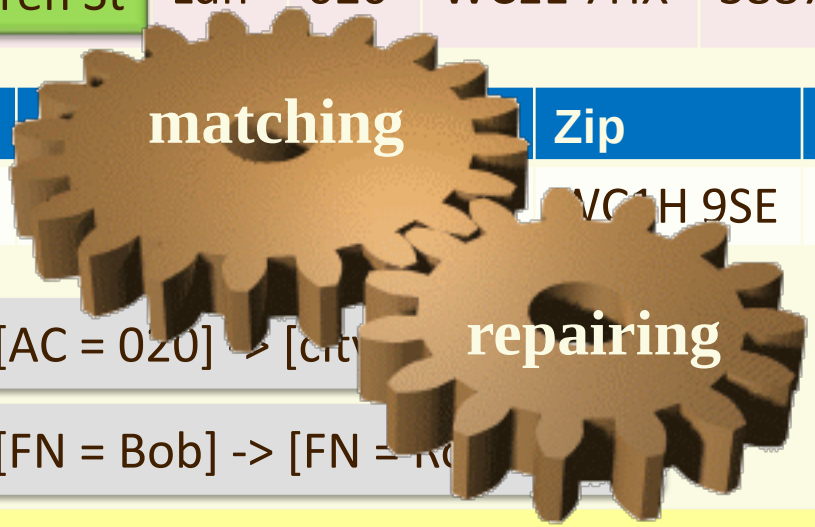
Unifying repairing and matching

Input data (tran)

FN	LN	St	City	AC	post	phn	item
Robert	Brady	5 Wren St	Ldn	020	WC1H 9SE	3887644	watch
Robert	Brady	5 Wren St	Ldn	020	WC1E 7HX	3887644	necklace

Master data (card)

FN	LN	Zip	tel
Robert	Brady	WC1H 9SE	3887644



1. tran([AC = 020] -> [city

2. tran([FN = Bob] -> [FN = Robert])

3. tran[LN, city, St, post] = card[LN,city, St, zip] ^
tran[FN] ≈ card[FN] -> tran[FN,phn] ⇔card[FN, tel]

4. tran([city, phn] -> [St, AC, post])

repairing helps matching

matching helps repairing

Repairing and matching operations should be interleaved

Summary and review

- Review questions:
  - What is data cleaning? Why bother?
  - What are the main approaches to cleaning data? Pros and cons?
  - Given a database D and a set C of FDs and INDs, can you always find a repair D’ of D?
  - What is the complexity of detecting errors in distributed data?
  - How to repair the data when the problem is NP-hard
- Research issues:
  - Algorithms for finding certain fixes for a set of input tuples
  - Incremental error detection in distributed data
  - XML data cleaning

Certain fix

How to fix the errors detected?

1. [AC=020] →[city=Ldn]
2. [AC=131] →[city=Edi]

Heuristic methods may not fix the erroneous t[AC], and worse still, may mess up the correct attribute t[city]

FN	LN	AC	phn	type	str	city	zip	item
Bob	Brady	020	079172485	2	501 Elm St	Ldn	EH7 4A	CD

Bob	Brady	131	079172485	2	501 Elm St.	Edi	EH7 4A	CD

Dependencies can detect errors, but do not tell us how to fix them

The quest for certain fixes

- ✓ 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 instead of dependencies
  - Editing rules tell us which attributes to change and how to change them; dynamic semantics
    Dependencies have static semantics: violation or not
  - Editing rules are defined on a master relation and an input relation – correcting errors with master data values
    Dependencies are only defined on input relation

Editing rules are a departure from data dependencies

Editing rules and m

1 – home phone
2 – mobile phone

certain

type=2

certain

	FN	LN	AC	phn	type	str	city	zip	item
t_1	Robert	Brad	131	07917248	2	501 Elm Row	Edi	EH7 4AH	CD

Input relation R

	FN	LN	AC	Hphn	Mphn	str	city	zip	DOB	gender
s_1	Robert	Brady	131	6884563	07917248	501 Elm Row	Edi	EH7 4AH	11/11/55	M
s_2	Mark	Smiths	020	6884563	075568485	20 Baker St.	Ldn	NW1 6XE	25/12/67	M

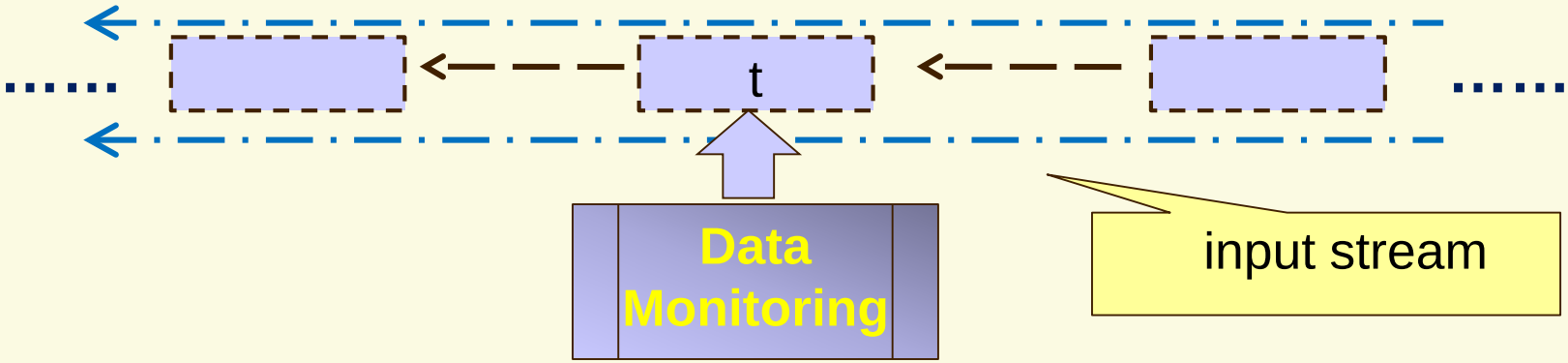
Master relation D_m

Certain regions: validated either by users or inference

Repairing: interact with users

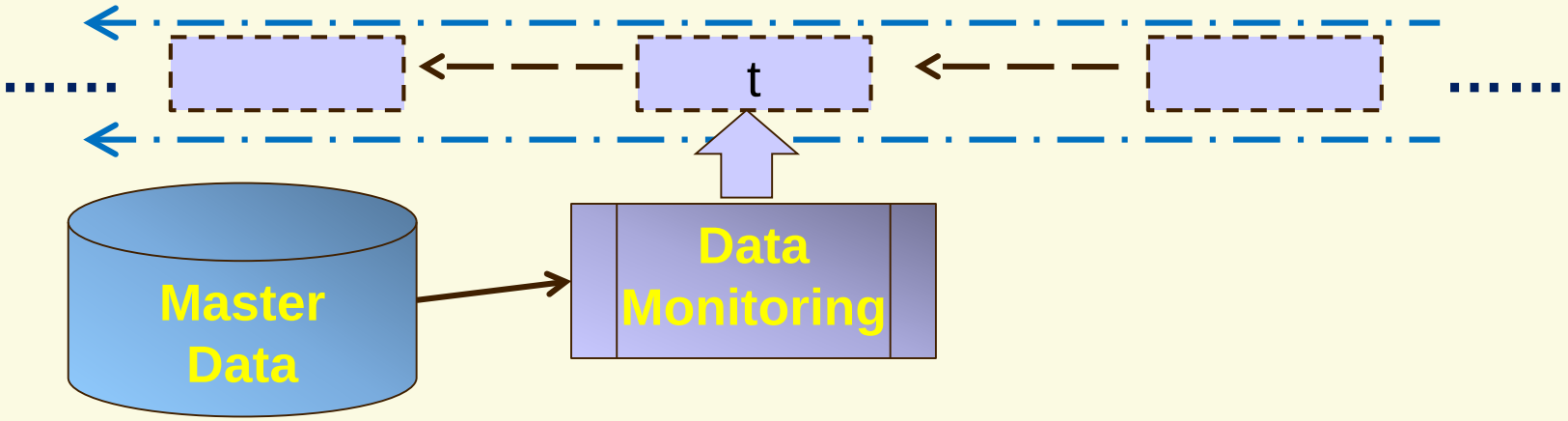
Applying editing rules does not introduce new errors

How do we find certain fixes?



far less costly to correct a tuple at the point of data entry than fixing it afterward.

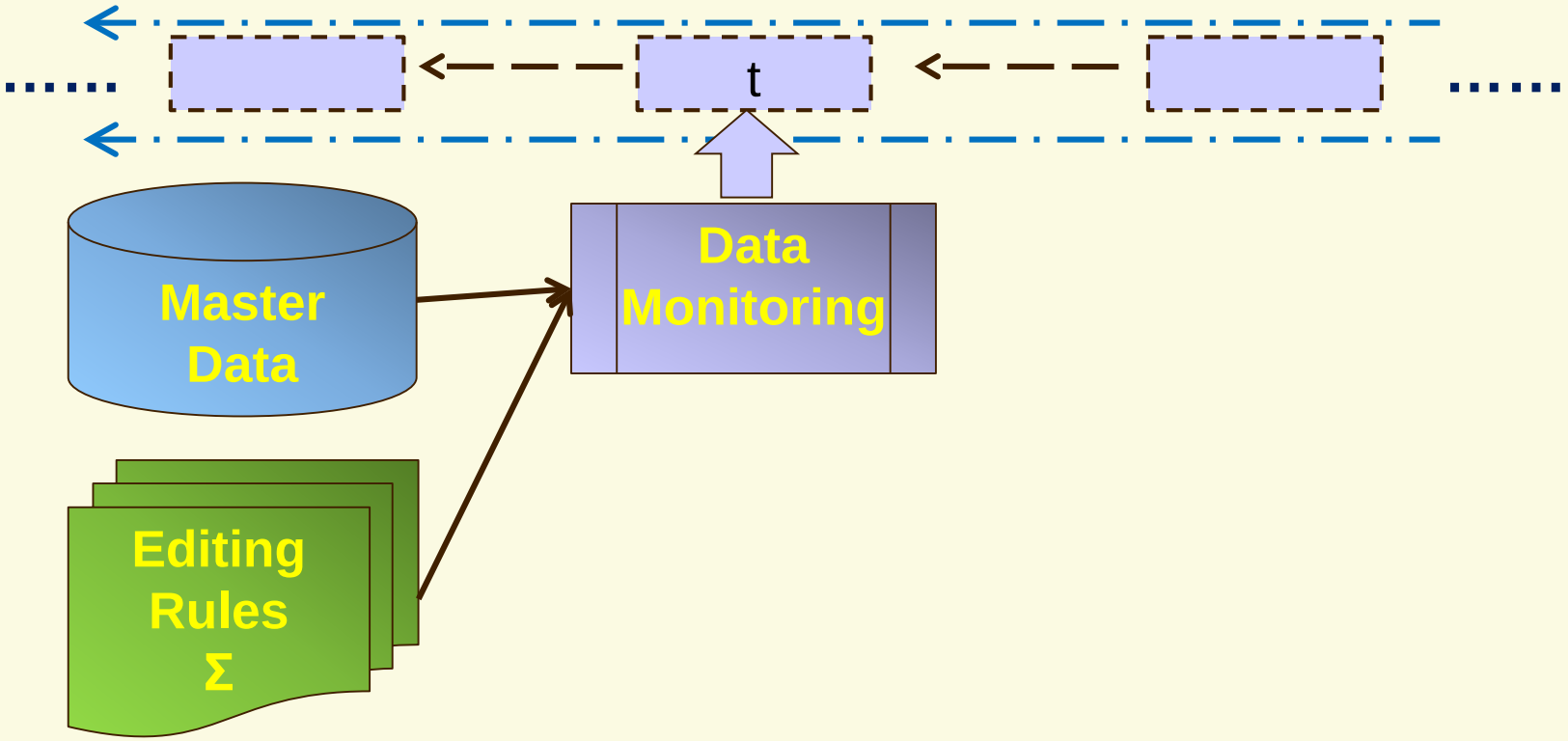
How do we find certain fixes?



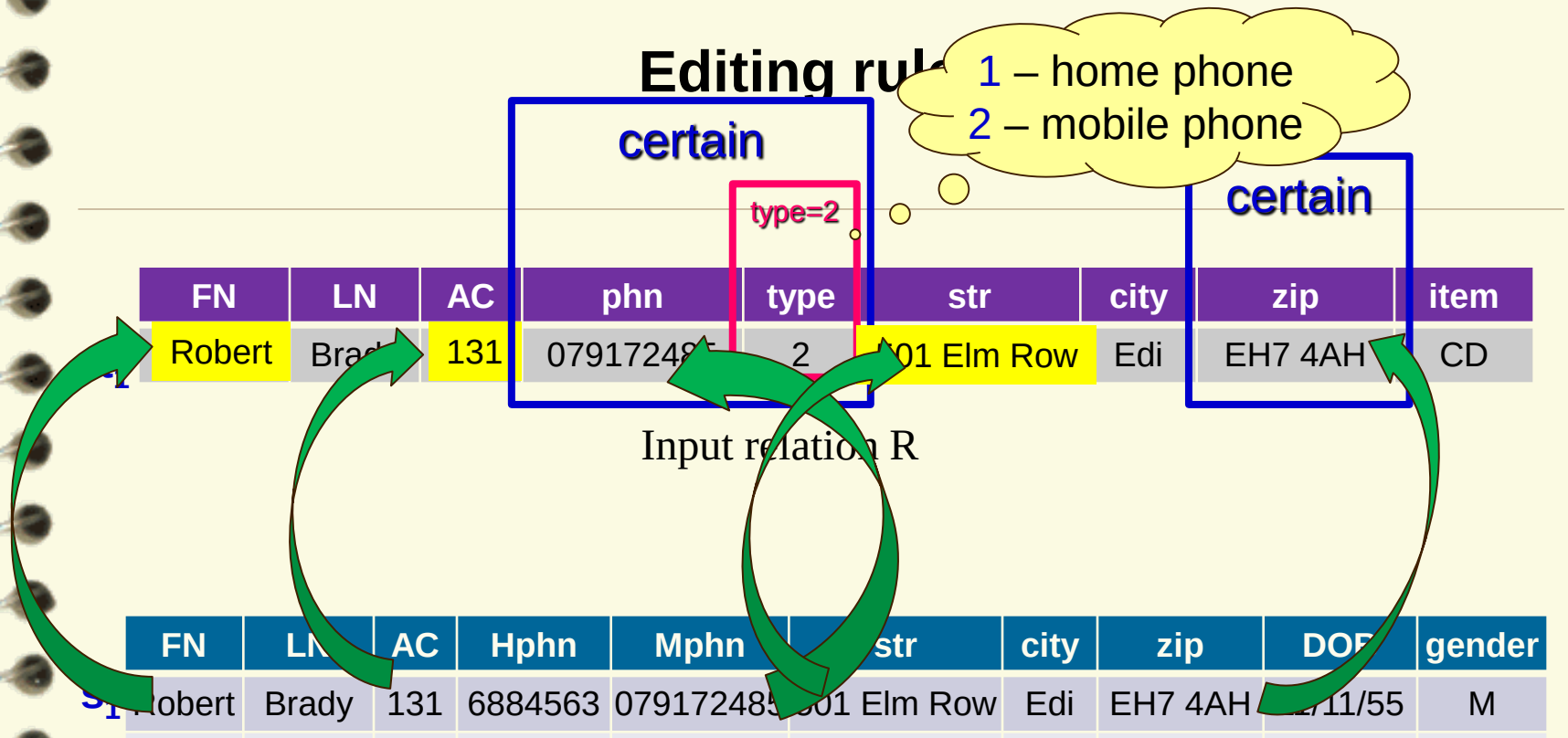
FN	LN	AC	Hphn	Mphn	str	city	zip	DOB	gender
Robert	Brady	131	6884563	079172485	501 Elm Row	Edi	EH7 4AH	11/11/55	M
Mark	Smiths	020	6884563	075568485	20 Baker St.	Ldn	NW1 6XE	25/12/67	M

Master relation D_m

How do we find certain fixes?



Editing rul



	FN	LN	AC	phn	type	str	city	zip	item
t_1	Robert	Brad	131	07917248	2	01 Elm Row	Edi	EH7 4AH	CD

Input relation R

	FN	LN	AC	Hphn	Mphn	str	city	zip	DOB	gender
s_1	Robert	Brady	131	6884563	07917248	01 Elm Row	Edi	EH7 4AH	11/11/55	M
s_2	Mark	Smiths	020	68						

- φ1: ((zip, zip) → (AC, str, city), tp1 = ( ))
- φ2: ((phn, Mphn) → (FN, LN), tp2[type] = (2))

Read: ***Towards certain fixes with editing rules and master data***

Certain fixes: editing rules, master data, certain region