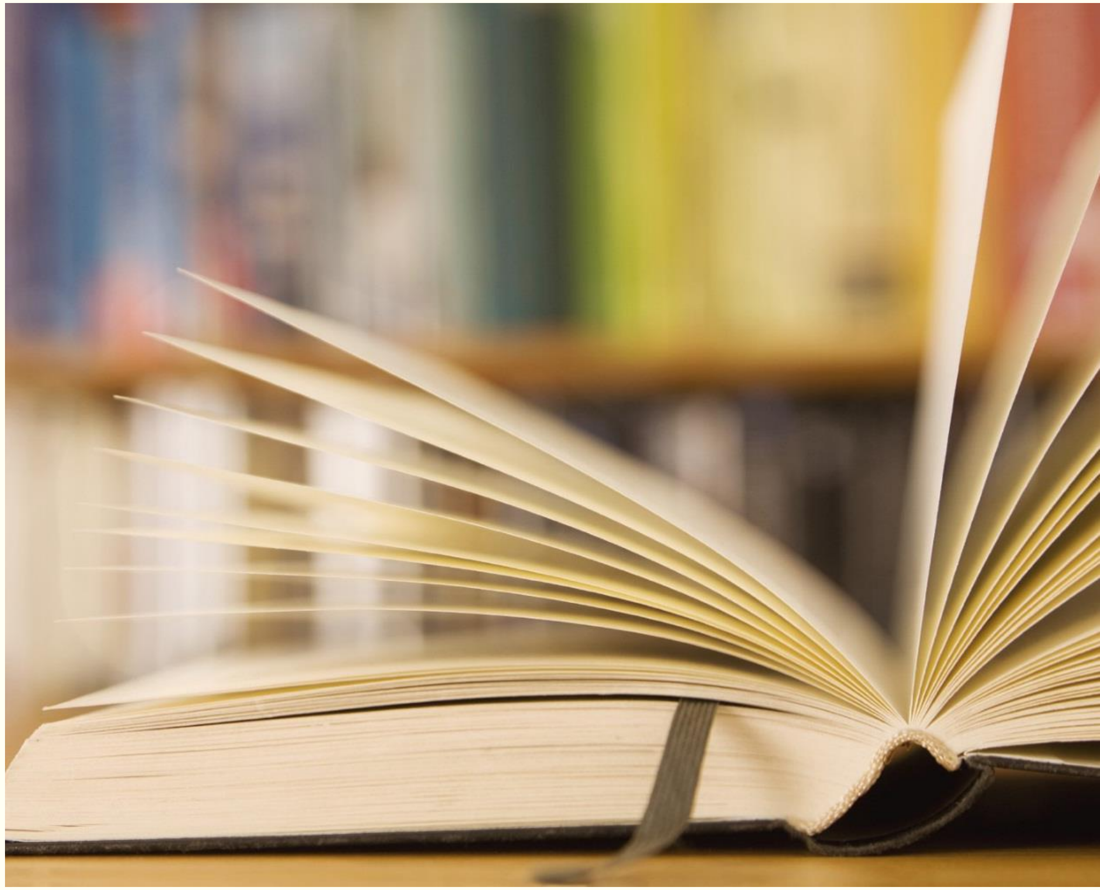
Data Quality



The Veracity of Big Data

- Data Quality Management: Overview
- Central Aspects of Data Quality
  - Data consistency
  - Entity resolution
  - Information completeness
  - Data currency
  - Data accuracy
  - Deducing the true values of objects in data fusion

Dependencies for Improving Data Quality

- Conditional functional dependencies (CFDs)
  - Syntax and semantics
- Conditional inclusion dependencies (CINDs)
  - Syntax and semantics
- Matching dependencies for record matching (MDs)
  - Syntax and semantics

Characterizing The Consistency Of Data

- One of the central technical problems for data consistency is how to tell whether the data is dirty or clean
- Integrity constraints (data dependencies) as data quality rules
- Inconsistencies emerge as violations of constraints
- Traditional dependencies
  - Functional dependencies
  - Inclusion dependencies
  - Denial constraints (a special case of full dependencies)
  - …
- Question: are these traditional dependencies sufficient?

Example: Customer Relation

- Schema:
  - Customer(country, area-code, phone, street, city, zip)
- Instance

country	AC	phone	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

- Functional dependencies (FDs)
  - Customer[country, area-code, phone] -> Customer[street, city, zip]
  - Customer[country, area-code] -> Customer[city]
- The database satisfies the FDs. Is the data consistent?

Capturing Inconsistencies in the Data

- Customer([country=44, zip] -> [street])
  - In UK, zip code uniquely determines the street.
  - The constraint may not hold for other countries
- It expresses a fundamental part of the semantics of the data
- It can NOT be expressed as a traditional FD
  - It does not hold on the entire relation;
  - Instead, it holds on tuples representing UK customers only

country	area-code	phone	street	city	zip
44	131	1234567	Mayfield	NYC	EH4 8LE
44	131	3456789	Crichton	NYC	EH4 8LE
01	908	3456789	Mountain Ave	NYC	07974

Two More Constraints

- Customer([country=44, area-code=131, phone] -> [street, zip, city=EDI])
  Customer([country=01, area-code=908, phone] -> [street, zip, city=MH])
  - In UK, if the area code is 131, the city has to be EDI
  - In US, if the area code is 908, the city has to be MH
- $t1, t2, t3$ violate these constraints
  - Refining Customer([country, area-code, phone] -> [street, zip, city])
  - Combining data values and variables

id	country	Area-code	phone	street	city	zip
t1	44	131	1234567	Mayfield	NYC	EH4 8LE
t2	44	131	3456789	Crichton	NYC	EH4 8LE
t3	01	908	3456789	Mountain Ave	NYC	07974

The Need For New Constraints

- Customer([country=44, zip]->[street])
  Customer([country=44, area-code=131, phone] -> [street, zip, city=EDI])
  Customer([country=01, area-code=908, phone] -> [street, zip, city=MH])
- They capture inconsistencies that traditional FDs cannot detect
  - Traditional constraints were developed for schema design, not for data cleaning!
- Data integration in real-life: source constraints
  - Hold on a subset of sources
  - Hold conditional on the integrated data
- They are NOT expressible as traditional FDs
  - Do NOT hold on the entire relation
  - Contain constant data values, besides logical variables

Conditional Functional Dependencies (CFDs)

- An extension of Traditional FDs (R: X-> Y, Tp)
  - X->Y: embedded traditional FD on R
  - Tp: A pattern tableau
    - Attributes: $X \cup Y$
    - Each tuple in Tp consists of constants and unnamed variable "_"
- Example: Customer([country=44, zip]->[street])
  - (Customer(country, zip -> street), Tp)
  - Pattern tableau Tp:

country	zip	street
44	_	_

Example CFDs

- Customer([country=44, zip]->[street])
  Customer([country=44, area-code=131, phone] -> [street, zip, city=EDI])
  Customer([country=01, area-code=908, phone] -> [street, zip, city=MH])
- The above can be represented as a single CFD
  - (Customer(country, area-code, phone -> street, city, zip), Tp)
  - Pattern Tableau Tp: One tuple for each constraints

country	area-code	phone	street	city	zip
44	131	_	_	Edi	_
01	908	_	_	MH	_
_	_	_	_	_	_

- CFDs subsume traditional FDs. Why?

Traditional FDs as a Special Case

- Traditional FD Example:
  - Customer[country, area-code] -> Customer[city]
- Corresponding CFD:
  - (Customer(country, area-code -> city), Tp)
  - Pattern Tableau Tp: A single tuple consisting of "_" only

country	area-code	city
_	_	_

Semantics of CFDs

- $a \approx b$ (a matches b) if
  - either a or b is _
  - both a and b are constants and a = b
- tuple t1 matches t2: $t1 \approx t2$
  - $(a, b) \approx (a, \_)$, but (a, b) does not match (a, c)
- DB satisfies (R: X->Y, Tp) iff for any tuple tp in the pattern tableau Tp and for any tuples t1, t2 in DB, if $t1[X] \ = \ t2[X] \approx tp[X]$, then $t1[Y] = \ t2[Y] \approx tp[Y]$
  - tp[X]: identifying the set of tuples on which the constraint tp applies, ie, $\{t|t[X] \approx tp[X]\}$
  - $t1[Y] \ = \ t2[Y] \approx tp[Y]$: enforcing the embedded FD, and the pattern of tp

Example: Violation of CFDs

- Example: Customer([country=44, zip]->[street])
  - (Customer(country, zip -> street), Tp)
  - Pattern tableau Tp

country	zip	street
44	_	_

id	country	area-code	phone	street	city	zip
t1	44	131	1234567	Mayfield	NYC	EH8 8LE
t2	44	131	3456789	Crichton	NYC	EH8 8LE
t3	01	908	3456789	Mountain Ave	NYC	07974

Tuples t1 and t2 violate the CFD

t1[country, zip] = t2[country, zip] $\approx$ tp[country, zip]

t1[street] $\neq$ t2[street]

The CFD applies to t1 and t2 since they match tp[country, zip]

Example: Violation of CFDs

country	area-code	city
44	131	Edi
01	908	MH
_	_	_

- (Customer(country, area-code -> city), Tp)

id	country	area-code	phone	street	city	zip
t1	44	131	1234567	Mayfield	NYC	EH8 8LE
t2	44	131	3456789	Crichton	NYC	EH8 8LE
t3	01	908	3456789	Mountain Ave	NYC	07974

Tuple t1 does not satisfy the CFD

t1[country, area-code] = t1[country, area-code] ≈ tp1[country, area-code]

t1[city] = t1[city]; however, t1[city] does not match tp1[city]

In contrast to traditional FDs, a single tuple may violate a CFD

Exercise

- (Customer(country, area-code, phone -> street, city, zip), Tp)

country	area-code	phone	street	city	zip
44	131	_	_	Edi	_
01	908	_	_	MH	_
_	_	_	_	_	_

- Violations? Why?

id	country	area-code	phon	street	city	zip
t1	44	131	1234567	Mayfield	Edi	EH4 8LE
t2	44	131	3456789	Mayfield	NYC	19082
t3	01	908	3456789	Mountain Ave	NYC	19082
t4	44	131	1234567	Chrichton	EDI	EH8 9LE

"Dirty" Constraints?

- A set of CFDs may be inconsistent!
  - E.g. (R(A->B), Tp)

id	A	B
tp1	_	b
tp2	_	c

- In any nonempty database DB, and for any tuple t in DB,
  - Tp1: t[B] must be b
  - Tp2: t[B] must be c
  - Inconsistent if b and c are different

- Another example: $\Sigma = \{\varphi 1, \varphi 2\}$
  - $\varphi 1 = (R(A \rightarrow B), Tp1)$
  - $\varphi 2 = (R(B \rightarrow A), Tp2)$

A	B
true	b
false	c

B	A
b	false
c	true

The Consistency Problem

- The consistency problem for CFDs is to determine, given a set Σ of CFDs, whether or not there exists a nonempty database DB that satisfies Σ ($\forall \varphi \in \Sigma$, DB satisfies φ)
- For traditional FDs, the consistency problem is not an issue
  - One can specify any FDs without worrying about their consistency
  - A set of CFDs may be inconsistent!
- Theorem: The consistency problem of CFDs is NP-Complete
  - Non-trivial: contrast this with the trivial consistency analysis of FDs

The Implication Problem

- The implication problem for CFDs is to determine, given a set Σ of CFDs and a single CFD φ, whether Σ implies φ, denoted by $\Sigma \vDash \varphi$
  - i.e. For any database DB, if DB satisfies Σ, then DB satisfies φ
- Example
  - $\Sigma = \{\varphi_1, \varphi_2\}$, $\varphi_1 = (R(A \to B), Tp1)$, $\varphi_2 = (R(B \to C), Tp2)$

A	B
_	b

B	C
_	c

  - $\varphi = (R(A \to C), Tp)$

A	C
a	c

Conditional Constraints for Data Cleaning

Example: Amazon Database

- Schema
  - Order (asin, title, type, price, country, country)
  - Book (asin, isbn, title, price, format)
  - CD (asin, title, price, genre)
- Instance:

Order

asin	title	type	price	country	county
a23	H. Porter	book	17.99	US	DL
a12	J. Denver	CD	7.94	UK	Reyden

Book

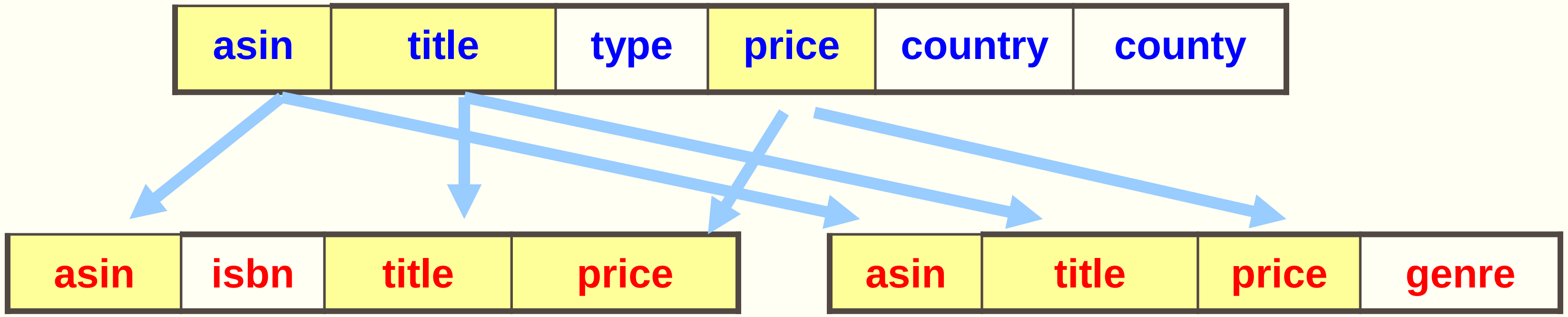
asin	isbn	title	price
a23	b32	Harry Porter	17.99
a56	b65	Snow white	7.94

CD

asin	title	price	genre
a12	J. Denver	17.99	country
a56	Snow White	7.94	a-book

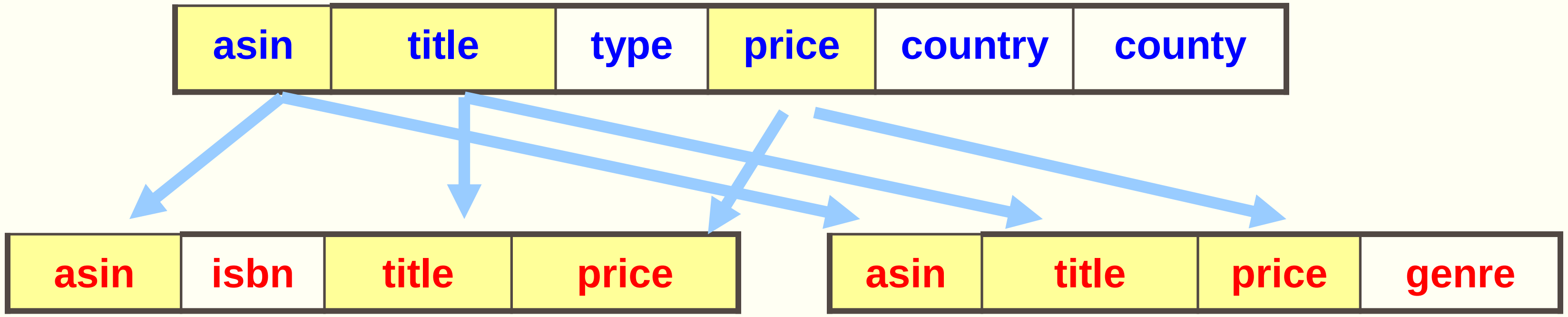
Schema Matching

- Inclusion dependencies from source to target



Order[asin, title, price] ⊆ Book[asin, title, price]   Order[asin, title, price] ⊆ CD[asin, title, price]

Schema Matching: Dependencies with Conditions



- Conditional Inclusion Dependencies (CIND):
  - $\text{Order}[\text{asin}, \text{title}, \text{price}; \text{type} = \text{book}] \subseteq \text{Book}[\text{asin}, \text{title}, \text{price}]$
    - $\text{Order}[\text{asin}, \text{title}, \text{price}] \subseteq \text{Book}[\text{asin}, \text{title}, \text{price}]$ holds only if type = book
  - $\text{Order}[\text{asin}, \text{title}, \text{price}; \text{type} = \text{CD}] \subseteq \text{CD}[\text{asin}, \text{title}, \text{price}]$
    - $\text{Order}[\text{asin}, \text{title}, \text{price}] \subseteq \text{CD}[\text{asin}, \text{title}, \text{price}]$ holds only if type = CD
- The constraints do not hold on the entire order table

Data Cleaning With Conditional Dependencies

- CIND1: $\text{Order}[\text{asin}, \text{title}, \text{price}; \text{type} = \text{book}] \subseteq \text{Book}[\text{asin}, \text{title}, \text{price}]$
- CIND2: $\text{Order}[\text{asin}, \text{title}, \text{price}; \text{type} = \text{CD}] \subseteq \text{CD}[\text{asin}, \text{title}, \text{price}]$

Order

id	asin	title	type	price	country	county
t1	a23	H. Porter	book	17.99	US	DL
t2	a12	J. Denver	CD	7.94	UK	Reyden

Book

asin	isbn	title	price
a23	b32	Harry Porter	17.99
a56	b65	Snow white	7.94

CD

asin	title	price	genre
a12	J. Denver	17.99	country
a56	Snow White	7.94	a-book

More on Data Cleaning

CD

asin	title	price	genre
a12	J. Denver	17.99	country
a56	Snow White	7.94	a_book

Book

asin	isbn	title	price	format
a23	b32	Harry Porter	17.99	Hard cover
a56	b65	Snow White	17.94	audio

- CIND: $CD[asin, title, price; genre = `a\_book'] \subseteq Book[asin, title, price; format = `audio']$
  - Inclusion dependency $CD[asin, title, price] \subseteq Book[asin, title, price]$ holds only if genre = `a-book` (when the CD is an audio book)
  - In addition, the format of the corresponding book must be "audio" – a pattern for the referenced tuple

Conditional Inclusion Dependencies (CINDs)

- $(R1[X; Xp] \subseteq R2[Y; Yp], Tp)$
  - $R1[X] \subseteq R2[Y]$: embedded traditional inclusion dependency from R1 to R2
  - Tp: a pattern tableau
    - Attributes: $Xp \cup Yp$
    - Tuples in Tp consist of constant and unnamed variable _
- Example:
  - CIND1: $\mathrm{Order[asin, title, price; type = book]} \subseteq \mathrm{Book[asin, title, price]}$
  - $(\mathrm{Order[asin, title, price; type]} \subseteq \mathrm{Book[asin, title, price; nil]}, \mathrm{Tp})$
    - Nil: Empty list

type
book

Traditional INDs As A Special Case

- $R1[X] \subseteq \mathrm{R2[Y]}$
  - X: [A1, A2, …, An]
  - Y: [B1, B2, …, Bn]
- As a CIND: $R1[X; \mathrm{nil}] \subseteq \mathrm{R2[Y; nil]}$
- What is the pattern tableau?

Exercise

- Express the following as CINDs:
  - $\text{Order}[\text{asin}, \text{title}, \text{price}; \text{type} = \text{CD}] \subseteq \text{CD}[\text{asin}, \text{title}, \text{price}]$
  - $\text{CD}[\text{asin}, \text{title}, \text{price}; \text{genre} = \grave{}\text{a\_book}'] \subseteq \text{Book}[\text{asin}, \text{title}, \text{price}; \text{format} = \grave{}\text{audio}']$

Semantics of CINDs

- DB = (DB1, DB2), where DBj is an istance of Rj, j = 1, 2
- DB satisfies $(R1[X; Xp] \subseteq R2[Y; Yp], Tp)$ iff for any tuples t1 in DB1 and any tuple tp in the pattern tableau Tp, if $t1[Xp] \approx tp[Xp]$, then there exist t2 in DB2 such that
  - $t1[Y] = t2[Y]$ (traditional INDs)
  - $t2[Yp] \approx tp[Yp]$ (matching the pattern tuple on Y, Yp)
- Patterns
  - $t1[Xp] \approx tp[Xp]$: identifying the set of R1 tuples on which tp applies: $\{t1 | t1[Xp] \approx tp[Xp]\}$
  - $t2[Yp] \approx tp[Yp]$: enforcing the embedded IND and the constraint specified by pattern Yp

Example

- $(\text{CD}[\text{asin}, \text{title}, \text{price}; \text{genre}] \subseteq \text{Book}[\text{asin}, \text{title}, \text{price}; \text{format}], Tp)$

genre	format
a-book	audio

- The following DB satisfies the CIND

asin	isbn	title	price	format
a23	b32	Harry Porter	17.99	Hard cover
a56	b65	Snow white	7.94	audio

asin	title	price	genre
a12	J. Denver	17.99	country
a56	Snow White	7.94	a-book

Exercise

- (Order[asin, title, price; type] ⊆ Book[asin, title, price; nil], Tp)

type
book

id	asin	title	type	price	country	county
t1	a23	H. Porter	book	17.99	US	DL
t2	a12	J. Denver	CD	7.94	UK	Reyden

asin	isbn	title	price
a23	b32	Harry Porter	17.99
a56	b65	Snow white	7.94

asin	title	price	genre
a12	J. Denver	17.99	country
a56	S. White	7.94	a-book

The Satisfiability Problem For CINDs

- The consistency problem for CINDs is to determine, given a set Σ of CINDs, whether or not there exists a nonempty database DB that satisfies Σ $(\forall \varphi \in \Sigma$, DB satisfies $\varphi)$
- Recall
  - Any set of traditional INDs is always consistent
  - For CFDs, the satisfiability problem is intractable
- Theorem: Any set of CINDs is always consistent!
- Despite the increased expressive power, the complexity of satisfiability analysis does not go up

The implication problem for CINDs

- The implication problem for CINDs is to decide, given a set Σ of CINDs and a single CIND φ, whether Σ implies φ, denoted by $\Sigma \vDash \varphi$
  - For traditional INDs, the implication problem is PSPACE-complete
  - For CINDs, the complexity does not hike up, to an extent
- Theorem, For CINDs containing no finite-domain attributes, the implication problem is PSPACE-complete
- Theorem: The implication problem of CINDs is EXPTIME-complete
  - In general settings, however, we have to pay a price

Record Matching

- To identify tuples from one or more unreliable sources that refer to the same real-world object.

Pairwise comparison of attributes via equality only does not work!

FN	LN	address	tel	DOB	gender
Mark	Smith	10 Oak St, EDI, EH8 9LE	3256777	10/27/97	M

the same person?

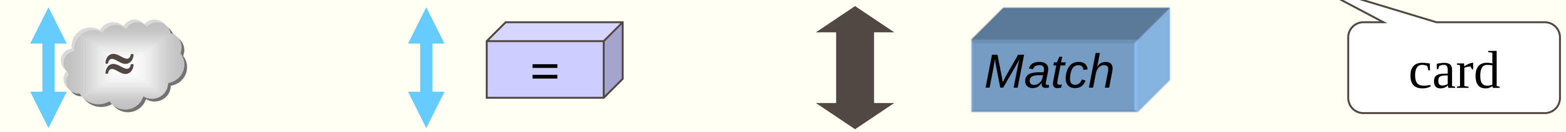
FN	LN	post	phn	when	where	amount
M.	Smith	10 Oak St, EDI, EH8 9LE	null	1pm/7/7/09	EDI	*$3,500*
…	…	…	…	…	…	…
Max	Smith	PO Box 25, EDI	3256777	2pm/7/7/09	NYC	$6,300

Record linkage, entity resolution, data deduplication, merge/purge

Matching Rules (Herndndez & Stolfo, 1995)

- IF card[LN, address] = trans[LN, post] AND card[FN] and trans[FN] are similar, THEN identify the two tuples

FN	LN	address	tel	DOB	gender
Mark	Smith	10 Oak St, EDI, EH8 9LE	3256777	10/27/97	M



FN	LN	post	phn	when	where	amount
M.	Smith	10 Oak St, EDI, EH8 9LE	null	1pm/7/7/09	EDI	*$3,500*
…	…	…	…	…	…	…
Max	Smith	PO Box 25, EDI	3256777	2pm/7/7/09	NYC	$6,300

trans

Dependencies for Record Matching

- card[LN, address] = trans[LN, post] $\wedge$ card[FN] $\approx$ trans[FN] $\rightarrow$ card[X] $\Leftrightarrow$ trans[Y]
- card[tel] = trans[phn] $\rightarrow$ card[address] $\Leftrightarrow$ trans[post]
- Identifying attributes (not necessarily entire records), across sources

X

card

FN	LN	address	tel	DOB	gender
Mark	Smith	10 Oak St, EDI, EH8 9LE	3256777	10/27/97	M

Y

trans

FN	LN	post	phn	when	where	amount
Max	Smith	PO Box 25, EDI	3256777	2pm/7/7/09	NYC	$6,300

$2^{(m*n)}$ configurations

Deducing New Dependencies From Given Rules

$$\text{card}[\text{LN},\text{address}] = \text{trans}[\text{LN},\text{post}] \wedge \text{card}[\text{FN}] \approx \text{trans}[\text{FN}] \rightarrow \text{card}[X] \Leftrightarrow \text{trans}[Y]$$

$$\text{card}[\text{tel}] = \text{trans}[\text{phn}] \rightarrow \text{card}[\text{address}] \Leftrightarrow \text{trans}[\text{post}]$$

deduction

$$\text{card}[\text{LN}, \text{tel}] = \text{trans}[\text{LN}, \text{phn}] \wedge \text{card}[\text{FN}] \approx \text{trans}[\text{FN}] \rightarrow \text{card}[X] \Leftrightarrow \text{trans}[Y]$$

card

FN	LN	address	tel	DOB	gender
Mark	Smith	10 Oak St, EDI, EH8 9LE	3256777	10/27/97	M

Match

Radically different

trans

FN	LN	post	phn	when	where	amount
Max	Smith	PO Box 25, EDI	3256777	2pm/7/7/09	NYC	$6,300

Matching Dependencies (MDs)

- $(R1[A1] \approx_1 R2[B1] \wedge \cdots \wedge R1[Ak] \approx_k R2[Bk]) \rightarrow R1[Z1] \Leftrightarrow R2[Z2]$
- R1[X], R2[Y]: entities to be identified
  - (Z1, Z2): lists of attributes in (X, Y), of the same length
  - $\approx_1$: similarity operator (edit distance, q-gram, jaro distance, …)
  - $\Leftrightarrow$: matching operator (identify two lists of attributes via updates)
- R1[X]: card[FN, LN, address], R2[Y]: trans[FN, LN, post]
  - $\text{card}[\text{LN}, \text{address}] = \text{trans}[\text{LN}, \text{post}] \wedge \text{card}[\text{FN}] \approx \text{trans}[\text{FN}] \rightarrow \text{card}[\text{X}] \Leftrightarrow \text{trans}[\text{Y}]$
  - $\text{card}[\text{tel}] = \text{trans}[\text{phn}] \rightarrow \text{card}[\text{address}] \Leftrightarrow \text{trans}[\text{post}]$
  - $\text{card}[\text{LN}, \text{tel}] = \text{trans}[\text{LN}, \text{phn}] \wedge \text{card}[\text{FN}] \approx \text{trans}[\text{FN}] \rightarrow \text{card}[\text{X}] \Leftrightarrow \text{trans}[\text{Y}]$

Dynamic Semantics

- $\varphi = (\mathrm{R1[A1]} \approx_1 \mathrm{R2[B1]} \wedge \cdots \wedge \mathrm{R1[Ak]} \approx_k \mathrm{R2[Bk]}) \rightarrow \mathrm{R1[Z1]} \Leftrightarrow \mathrm{R2[Z2]}$
- (D1, D2) satisfies φ iff for all $(\mathrm{t1}, \mathrm{t2}) \in \mathrm{D1}$
  - if $\mathrm{t1[A1]} \approx_1 \mathrm{t2[B1]} \wedge \cdots \wedge \mathrm{t1[Ak]} \approx_k \mathrm{t2[Bk]}$ in D1
  - Then $(\mathrm{t1}, \mathrm{t2}) \in \mathrm{D2}$ and $\mathrm{t1[Z1]} = \mathrm{t2[Z2]} \in D2$
- If (t1, t2) match the LHS, then their RHS are updated and equalized

tel	address	…
3256777	10 Oak St, EDI	

phn	post	…
3256777	PO Box 25, EDI	

D1

tel	address	…
3256777	10 Oak St, EDI, EH8 9LE	

phn	post	…
3256777	10 Oak St, EDI, EH8 9LE	

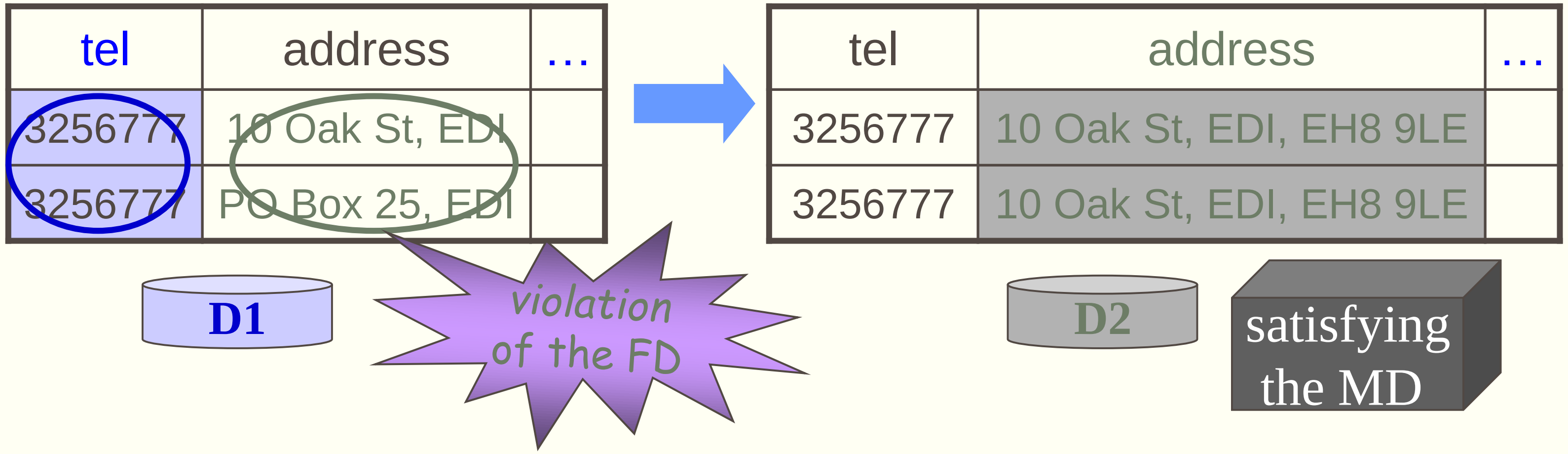
D2

An Extension Of Functional Dependencies (Fds)?

MD: card[tel] = trans[phn] $\rightarrow$ card[address] $\Leftrightarrow$ trans[post]

FD: tel $\rightarrow$ address

- similarity operators vs. equality (=) only
- across different relations (R1, R2) vs. on a single relation
- dynamic semantic (matching operator) vs. static semantics



Summary

- What are CFDs? CINDs? Why do we need new constraints?
- What is the consistency problem? Complexity?
- What is the implication problem? Inference system? Sound and complete?
- What is record matching? Why bother?
- What are matching rules?
- A practical question: how to discover these constraints? A learning/Mining problem.

Reading List

- W. Fan, F. Geerts, X. Jia and A. Kementsietsidis. Conditional Functional Dependencies for Capturing Data Inconsistencies, TODS, 33(2), 2008.
- L. Bravo, W. Fan. S. Ma. Extending dependencies with conditions. VLDB 2007.
- W. Fan, J. Li, X. Jia, and S. Ma. Dynamic constraints for record matching, VLDB, 2009.
- L. E. Bertossi, S. Kolahi, L.Lakshmanan: Data cleaning and query answering with matching dependencies and matching functions, ICDT 2011. http://people.scs.carleton.ca/~bertossi/papers/matchingDC-full.pdf
- F. Chiang and M. Miller, Discovering data quality rules, VLDB 2008. http://dblab.cs.toronto.edu/~fchiang/docs/vldb08.pdf
- L. Golab, H. J. Karloff, F. Korn, D. Srivastava, and B. Yu, On generating near-optimal tableaux for conditional functional dependencies, VLDB 2008. http://www.vldb.org/pvldb/1/1453900.pdf