Incomplet$\genfrac{}{}{0pt}{}{\text{e}}{\text{a}}$ In?ormation

"Informationssysteme" SS 2007

Dan Olteanu, Lehrstuhl für Informationssysteme, Universität des Saarlandes

Why Can Information Be Incomplete? (1)

Social Security Number:	785
Name:	Smith
Marital Status:	(1) single ☒ (2) married ■
	(3) divorced ☐ (4) widowed ☐

Social Security Number:	185
Name:	Brown
Marital Status:	(1) single ☐ (2) married ☐
	(3) divorced ☐ (4) widowed ☐

Why Can Information Be Incomplete? (2)

Single-source problems

- schema level (lack of integrity constraints, poor schema design)
  uniqueness, referential integrity
- instance level (data entry errors)
  misspellings, redundancy/duplicates, contradictory values

Multi-source problems

- schema level (heterogeneous data models and schema design)
  naming and structural conflicts
- instance level (overlapping, contradicting, and inconsistent data)
  inconsistent aggregating, inconsistent timing

Why Can Information Be Incomplete? (3)

Single-source problems at schema level

Scope	Problem	Dirty Data	Remarks
Attribute	illegal	bdate=30.13.70	out-of-range value
Record	dependency	age=22, bdate=12.02.70	age=now - bdate
Record type	uniqueness	emp1(SSN=1),emp2(SSN=1)	SSN is unique
Source	ref. integrity	emp(SSN=007)	007 not defined

Why Can Information Be Incomplete? (4)

Single-source problems at instance level

Scope	Problem	Dirty Data
Attribute	missing value	null
	misspelling	Lizpig
	cryptic values	DB Prog.
	embedded values	name="Joe New York"
	misfielded values	city=Germany
Record	dependencies	city=SB, code=85764
Record type	transpositions	"D.Olteanu", "Olteanu D."
	duplicates	emp("Dan Olteanu"), emp("D.Olteanu")
	contradictions	emp(Olteanu,Koch), emp(Olteanu,Bock)

How to Cope with Incompleteness? (Approach 1)

Remove all instances (= worlds) that do not satisfy particular criteria.
The hope is to get one (clean) instance in the end.
Data Cleaning deals with detecting and removing errors and inconsistencies from data in order to improve the quality of the data.
It usually consists of the following steps:

1. Data analysis (to detect the kind of occurring errors)
   data mining
2. definition of transformation and mapping rules
   declarative SQL-based languages
3. verification
4. transformation
5. backflow of cleaned data

Data cleaning is a complex semiautomatic approach to deal with incomplete data.

How to Cope with Incompleteness? (Approach 2)

Complementary approach: Provide support for

1. efficient (=succinct) representation of (possibly infinte) sets of worlds.
2. define processing (query evaluation, dependency chasing) on such succinct representations.

We further discuss this approach.

Example of Incomplete Information

Persons	Name	Salary	Room	Phone
	DAO	40K	228	?
	LRA	10K	?	?
	CEK	?	226	57328

? usually represented as **null** value in existing RDBMSs

SQL supports NULL values with constructs like IS (NOT) NULL.

Compare the answers to the following two queries
Q_1: SELECT FROM Persons WHERE Room $>$ 226;
Q_2: SELECT FROM Persons WHERE Room $>$ 226 OR room IS NULL;

Example of Incomplete Information (cont'd)

There are different types of nulls (?).

1. existing unknown values, e.g., DAO's phone or CEK's salary
2. nonexisting values, e.g., LRA's phone
3. no information is known about, e.g., LRA's room number

Persons	Name	Salary	Room	Phone
	DAO	40K	228	?
	LRA	10K	?	
	CEK	?	226	57328

We consider next nulls of the first two kinds.

Completeness versus Incompleteness (1)

A relation with null values encodes a set of possible *worlds*.

Persons	Name	Salary	Room	Phone
	DAO	40K	228	57332
	LRA	10K	MPIRS-1	
	CEK	400K	226	57328

Persons	Name	Salary	Room	Phone
	DAO	40K	228	57332
	LRA	10K	MPIRS-2	
	CEK	500K	226	57328

.. and so on.

There is an infinite amount of possible worlds!!!

Represent intensionally the set of possible worlds

Representation Systems for Incomplete Information

What is a representation system?

System to represent set of alternatives or *possible worlds*.

- World = (complete) database.
- Representation T (usually called *Table*)
- Function *rep* mapping T to the set of *possible worlds*.

Query evaluation under *possible world semantics*

$$
\begin{tikzcd}
T \arrow[r, "q"] \arrow[d, "rep"'] & \overline{q}(T) \arrow[d, "rep"] \\
\boxed{\{\mathcal{A}_1, \ldots, \mathcal{A}_n\}} \arrow[r, "q"] & \boxed{\{q(\mathcal{A}_1), \ldots, q(\mathcal{A}_n)\}}
\end{tikzcd}
$$

Strong Representation Systems

Language $\mathcal{L}$ (e.g., relational algebra) and *table* T with $rep(T)$

- For a query $q \in \mathcal{L}$, collect the set of possible answers

$$\boxed{q(rep(T)) = \{q(I) \mid I \in rep(T)\}}$$

- *represent*! $q(rep(T))$ as a table $\overline{q}(T)$

$$\boxed{rep(\overline{q}(T)) = q(rep(T))}$$

If T is any table in a representation system τ and q any query in $\mathcal{L}$, then

$$\boxed{\tau \text{ is a } \textit{strong representation system} \text{ for } \mathcal{L}}$$

Weak Representation Systems

$\mathcal{L}$-Equivalence $\equiv_{\mathcal{L}}$ of Incomplete Databases
Language $\mathcal{L}$, two incomplete databases $\mathcal{I}$ and $\mathcal{J}$.

$$\boxed{\mathcal{I} \equiv_{\mathcal{L}} \mathcal{J} \Leftrightarrow \forall q \in \mathcal{L} : \bigcap\{q(I) \mid I \in \mathcal{I}\} = \bigcap\{q(J) \mid J \in \mathcal{J}\}}$$

$\bigcap\{q(J) \mid J \in \mathcal{J}\}$ is the *certain* answer (or the set of *sure* answer tuples)
$\mathcal{I}$ and $\mathcal{J}$ are equivalent if all we can ask for is the certain answer of $\mathcal{L}$-queries.

If T is any table of a representation system τ and q any query in $\mathcal{L}$, then

$$\boxed{\tau \text{ is a } \textit{weak representation system} \text{ for } \mathcal{L} \Leftrightarrow rep(\overline{q}(T)) \equiv_{\mathcal{L}} q(rep(T))}$$

Corollary: If a system is strong for $\mathcal{L}$, then it is also weak for $\mathcal{L}$.

(Codd) Tables

- Codd tables = Finite relations, where tuples can contain variables
- A variable can occur at most once per entire table
- A Codd table T represents the incomplete database (set of possible worlds)

$$\boxed{rep(T) = \{\nu(T) \mid \nu \text{ is a valuation of the variables in } T\}}$$

R	A	B	C
	0	1	x
	y	z	1
	2	0	v

contains

R	A	B	C
	0	1	2
	2	0	1
	2	0	0

R	A	B	C
	0	1	2
	3	0	1
	2	0	5

...

Querying Codd tables: Selection

R	A	B	C
	0	1	x
	y	z	1
	2	0	v

$\xrightarrow{\sigma_{A=3}(R)}$

R	A	B	C
	3	z	1

R	A	B	C

There is no Codd table representing the set of all possible answers!
But there is a (empty) Codd table representing the *certain* answer!

$\boxed{\text{Codd tables form no strong representation system for selection}}$

Querying Codd tables: Projection

R	A	B	C
	0	1	x
	y	z	1
	2	0	v

$\xrightarrow{\pi_A(R)}$

$\pi_A(R)$	A
	0
	y
	2

$\boxed{\text{Codd tables form a strong representation system for projection}}$

Querying Codd tables: Product and Join

R	A	B	C
	0	1	x
	y	z	1
	2	0	v

S	D
	0
	1

$\xrightarrow{R \times S}$

$R \times S$	A	B	C	D
	0	1	x	0
	0	1	x	1
	y	z	1	0
	y	z	1	1
	2	0	v	0
	2	0	v	1

A variable can appear only once in a Codd table!

Codd tables form no strong representation system for product and join

Querying Codd tables: Union

R	A	B	C
	0	1	x
	y	z	1
	2	0	v

S	A	B	C
	1	1	x

$\xrightarrow{R \cup S}$

$R \cup S$	A	B	C
	0	1	x
	y	z	1
	2	0	v
	1	1	x

A variable can appear only once in a Codd table!

Codd tables form no strong representation system for union

Querying Codd tables: Difference

R	A	B	C
	0	1	x
	y	z	1
	2	0	v

S	A	B	C
	2	0	0

$\xrightarrow{R-S}$

$R-S$	A	B	C
	0	1	x
	y	z	1
	2	0	?

The value of ? can be anything but 0!

$\boxed{\text{Codd tables form no strong representation system for difference}}$

Certain Answers for Codd tables

For a table T and a query q, the certain answer is

$$\boxed{sure(q, T) = \bigcap\{q(I) \mid I \in rep(T)\}}.$$

- Sure facts appear in the answer for *every* possible world.
- Compute $sure(q, T)$ by dropping all tuples with variables in $q(rep(T))$.
- For our Codd table T, $sure(\sigma_{A=3}(R), T) = \emptyset$, thus representable as Codd table!
- Representing *only* the sure answer tuples is not sufficent!

R	A	B	C
	0	1	x
	y	z	1
	2	0	v

- Consider $q = \sigma_{A=2}(R)$ and $q' = \pi_{AB}(R)$
- Then, $sure(q, T) = \emptyset \Rightarrow q'(sure(q, T)) = \emptyset$
- But, $sure(q'(q(rep(T)))) = \{(2, 0)\} \neq \emptyset$
- $\Rightarrow$ non-compositional query semantics!

How Weak are Codd tables? (1)

R	A	B	C
	0	1	x
	y	z	1
	2	0	v

- Consider again $q = \sigma_{A=2}(R)$ and $q' = \pi_{AB}(R)$
- Choose projection $\overline{q}' = q'$ and selection $\overline{q}_\theta$ such that $\overline{q}_\theta(T) = \{t \mid t \in T, \forall$ valuations of vars in t $\mu : \theta(\mu(t))\}$
- Then, $\overline{q}(T) = \{(2,0,v)\}$ and $\overline{q' \circ q}(T) = \{(2,0)\}$.

Codd tables form a weak representation system for selections and projections

How Weak are Codd tables? (2)

- Consider $q = \pi_{AC}(R) \bowtie \pi_B(R)$
- Suppose there is a table W such that $rep(W) \equiv_{SPJ} q(rep(T))$
- Consider $q' = \pi_{AC}(\pi_{AB}(R) \bowtie \pi_{BC}(R))$; $q \circ q' = \pi_A(R) \times \pi_C(R)$
- Show that $rep(\overline{q}'(W)) \not\equiv_{SPJ} q'(rep(W))$
- Equivalently, show that $sure(q', W) \neq sure(q \circ q', T)$
- Idea: for each valuation of vars in W, $(a', c) \in sure(q \circ q', T)$ but there are valuations such that $(a', c) \notin sure(q', W)$.

R	A	B	C
	a	x	c
	a'	x'	c'

$sure(q \circ q', T)$	A	C
	a	c
	a'	c
	a	c'
	a'	c'

$\boxed{\text{Codd tables form no weak representation system for SPU/SPJ}}$

Or-set Relations

Codd tables, where each variable takes values from a finite domain.

Census	SSN	Name	Marital Status
	{ 185, 785 }	Smith	{ 1, 2 }
	{ 185, 186 }	Brown	{ 1, 2, 3, 4 }

Number of represented worlds: $2 \cdot 1 \cdot 2 \cdot 2 \cdot 1 \cdot 4 = 32$.

C	SSN	Name	MS
	185	Smith	1
	185	Brown	1

C	SSN	Name	MS
	185	Smith	1
	185	Brown	2

C	SSN	Name	MS
	185	Smith	1
	185	Brown	3

C	SSN	Name	MS
	185	Smith	1
	185	Brown	4

and so on.

(Naive) v-tables

v-tables are Codd tables, where a variable can occur several times.

R	A	B	C
	0	1	x
	x	z	1
	2	0	v

v-tables form a *weak representation system* for positive relational algebra

Proof Idea

- treat variables in v-tables as constants
- perform standard evaluation on the table

Querying v-tables

R_1	A	B	C
	0	1	x
	x	z	1
	2	0	v

R_2	A	B	C
	1	1	x
	x	z	1

R_3	C	D
	1	1
	x	z

$\overline{\pi_B(R_1)}$	B
	1
	z
	0

$\overline{R_2 \bowtie R_3}$	A	B	C	D
	1	1	x	z
	x	z	1	1

$\overline{R_1 \cup R_2}$	A	B	C
	0	1	x
	x	z	1
	2	0	v
	1	1	x

$\overline{\sigma_{C=1}(R_3)}$	C	D
	1	1

(Conditional) c-tables

c-tables are triples (T, Φ_T, ϕ), where

- T is a v-table,
- Φ_T is a *global* condition,
- ϕ associates a local condition ϕ_t to each tuple t of T.

$rep(T) = \{\mathcal{A} \mid \exists$ valuation ν: $\nu(\Phi_T)$, $\mathcal{A} = \{\nu(t) \mid t \in T, \nu(\phi_t)\}\}$.

Condition = conjunct of (in-)equality atoms, e.g., $x = \mathrm{c}$, $x = y$, $x \neq \mathrm{c}$, $x \neq y$.

- *true* represented as $x = x$ (or simply omitted), false represented as $x \neq x$

c-table Example (1)

R_1	Student	Course	
			$x \neq math \wedge x \neq CS$
	Sally	*math*	$z = 0$
	Sally	*CS*	$z \neq 0$
	Sally	x	
	Alice	*bio*	$z = 0$
	Alice	*math*	$x = physics \wedge t = 0$
	Alice	*physics*	$x = physics \wedge t \neq 0$

c-table Example (2)

R_2	Student	Course	
			true
	Sally	*math*	$z = 0$
	Sally	*CS*	$z \neq 0$
	Sally	x	$x \neq \mathit{math} \wedge x \neq \mathit{CS}$
	Alice	*bio*	$z = 0$
	Alice	*math*	$x = \mathit{physics} \wedge t = 0$
	Alice	*physics*	$x = \mathit{physics} \wedge t \neq 0$

R_2 is R_1, where the global condition becomes local to the third tuple.

$$\boxed{R_1 \neq R_2}$$

How Strong are c-tables? (1)

$\boxed{\text{c-tables form a } \textit{strong representation system} \text{ for relational algebra}}$

Proof Idea

- projection is standard
- selection adds new conjuncts to the local condition
- union is standard
- difference adds a huge conjunct C_t to the local condition of each tuple from the first table
  C_t states that t does not match any tuple from the second table
- ...

How Strong are c-tables? (2)

T_1	B	C
	x	c

T_2	B	C	
	y	c	y=b
	z	w	

T_3	A	B
	a	y

$\overline{\pi_B(T_2)}$	B	
	y	$y = \text{b}$
	z	

$\overline{T_1 \cup T_2}$	B	C	
	x	c	
	y	c	$y = \text{b}$
	z	w	

$\overline{T_1 \bowtie T_3}$	A	B	C	
	a	y	c	$y = x$

$\overline{\sigma_{B=b}(T_1 \bowtie T_3)}$	A	B	C	
	a	y	c	$y = \text{b} \wedge y = x$

$\overline{T_1 - T_2}$	B	C	
	x	c	$y \neq \text{b} \wedge x \neq z$
	x	c	$y \neq \text{b} \wedge w \neq \text{c}$
	x	c	$y = \text{b} \wedge x \neq \text{b} \wedge x \neq z$
	x	c	$y = \text{b} \wedge x \neq \text{b} \wedge w \neq \text{c}$

How Strong are c-tables? (3)

$\boxed{\text{c-tables can represent answers of transitive closure queries}}$

T	A	B
	a	b
	x	c
	c	d

$\overline{tc}(T)$	A	B	
	a	b	
	x	c	
	c	d	
	a	c	$x = \text{b}$
	x	d	
	c	c	$x = \text{d}$
	a	d	$x = \text{b}$

Overview of Representation Systems

System	Is Weak For..	Is Strong For..
Codd tables	PS	P
v-tables	PS^+UJ	PU
c-tables	PSUJD	PSUJD

P = Projection, S = Selection, S^+ = pos S, U = Union, J = Join, D = Difference.

Decision Problems for Representation Systems

Decision Problems

Input	Representation system $\mathcal{W}$, instance $I = (R^I)$, tuple t	
Problems	Tuple Possibility:	$\exists \mathcal{A} \in rep(\mathcal{W}) : t \in R^{\mathcal{A}}$
	Tuple Certainty:	$\forall \mathcal{A} \in rep(\mathcal{W}) : t \in R^{\mathcal{A}}$
	Instance Possibility:	$\exists \mathcal{A} \in rep(\mathcal{W}) : R^I = R^{\mathcal{A}}$
	Instance Certainty:	$\forall \mathcal{A} \in rep(\mathcal{W}) : R^I = R^{\mathcal{A}}$
	Tuple Q-Possibility (query Q fixed):	$\exists \mathcal{A} \in rep(\mathcal{W}) : t \in Q(\mathcal{A})$
	Tuple Q-Certainty (query Q fixed):	$\forall \mathcal{A} \in rep(\mathcal{W}) : t \in Q(\mathcal{A})$
	Instance Q-Possibility (query Q fixed):	$\exists \mathcal{A} \in rep(\mathcal{W}) : R^I = Q(\mathcal{A})$
	Instance Q-Certainty (query Q fixed):	$\forall \mathcal{A} \in rep(\mathcal{W}) : R^I = Q(\mathcal{A})$

Decisions for c-tables

R_1	Student	Course	
			$x \neq math \wedge x \neq CS$
	Sally	*math*	$z = 0$
	Sally	*CS*	$z \neq 0$
	Sally	x	
	Alice	*bio*	$z = 0$
	Alice	*math*	$x = physics \wedge t = 0$
	Alice	*physics*	$x = physics \wedge t \neq 0$

- Which of the following tuples is possible/certain?
  (Alice,bio), (Sally,math), (Sally,bio), (Sally,agriculture), (Banana,bio)
- Which of the following tuples is $\pi_{Student}(R)$-certain? (Sally), (Alice)
- Which of the following instances is possible/certain?
  $\emptyset$, {(Alice,bio),(Sally,CS)}, {(Sally,CS),(Sally,math)}

Complexity of Decision Problems

	v-tables	c-tables
Tuple Possibility	PTIME	NP-compl[1].
Tuple Certainty	PTIME	coNP-compl[2].
Instance Possibility	NP-compl.	NP-compl.
Instance Certainty	PTIME	coNP-compl.
Tuple Q-Possibility	NP-compl.	NP-compl.
positive relational algebra	PTIME	NP-compl.
Tuple Q-Certainty	coNP-compl.	coNP-compl.
positive relational algebra	PTIME	coNP-compl.
Instance Q-Possibility	NP-compl.	NP-compl.
Instance Q-Certainty	coNP-compl.	coNP-compl.
positive relational algebra	PTIME	coNP-compl.

Footnotes: Simple reductions of SAT (for 1) and 3DNF-tautology (for 2).

Chasing Dependencies on Representation Systems

A Short Reminder on Dependencies

Consider a schema $U = (ABC)$ and a relation R over U.
R satisfies the functional dependency (FD) $A \rightarrow B$ if
$\forall x, y, z, y_1, z_1 (R(x, y, z) \wedge R(x, y_1, z_1) \Rightarrow y = y_1)$.

R satisfies the multivalued dependency (MVD) $A \twoheadrightarrow B$ if
$\forall x, y, z, y_1, z_1 (R(x, y_1, z) \wedge R(x, y, z_1) \Rightarrow R(x, y, z))$.
In other words, $A \twoheadrightarrow B$ ensures the new R becomes $\pi_{AC}(R) \bowtie \pi_{AB}(R)$.

Chasing Dependencies on World-Sets

Dependencies can be used to

1. eliminate inconsistent worlds. This leads to less worlds thus more information.
2. change inconsistent worlds such that they become consistent. This preserves the number of worlds, yet new tuples can be added or existing tuples can be dropped in the inconsistent worlds.

For FDs (and equality-generating dependencies in general):

- **we consider here the first semantics**, thus drop the inconsistent worlds.
- the second semantics leads to the notion of repairs wrt FDs:
  We replace an inconsistent world by a set of (minimal) consistent repairs.

For MVDs (and tuple-generating dependencies in general):

- **we consider here the second semantics**, thus add tuples to make the world consistent.
- the first semantics would eliminate the inconsistent worlds.

Example of Chasing Dependencies on World-Sets

Dependencies may help to eliminate inconsistent worlds from the set of possible worlds. An incomplete database $\mathcal{I}$ with a set Σ of dependencies represents

$$\boxed{\{chase(I, \Sigma) \mid I \in \mathcal{I} \text{ and the chase of } I \text{ by } \Sigma \text{ succeeds}\}}$$

Chasing $\Sigma = \{A \twoheadrightarrow B, B \rightarrow A\}$ on $\mathcal{I} = \{I_1, I_2, I_3\}$ leads to $\mathcal{I} = \{J_1, J_2\}$.

I_1	A	B	C
	a	b	c
	a	b'	c'

I_2	A	B	C
	e	f	g
	e	f'	g'
	e	f	g'
	e	f'	g

I_3	A	B	C
	a	b	c
	g	b	h

J_1	A	B	C
	a	b	c
	a	b'	c'
	a	b	c'
	a	b'	c

J_2	A	B	C
	e	f	g
	e	f'	g'
	e	f	g'
	e	f'	g

Chasing Dependencies on v-tables

A v-table can be seen as the core (without the head) of a tableau query!

R	A	B	C
	0	1	x
	x	z	1
	2	1	2

Chase $B \longrightarrow C$ on R

R'	A	B	C
	0	1	2
	2	z	1
	2	1	2

Chasing Dependencies on c-tables

$\Sigma = \{A \twoheadrightarrow B, C \rightarrow D\}$, c-tables T_1 and T_2. Then, $\boxed{chase_\Sigma(rep(T_1)) = rep(T_2)}$.

T_1	A	B	C	D
	a	b	c	d
	x	e	y	g
	a	b	c	z

T_2	A	B	C	D	
					c=c $\Rightarrow z =$ d
	a	b	c	d	
	x	e	y	g	
	a	b	y	g	$x =$ a
	a	e	c	z	$x =$ a

- FDs add Horn formulas to the global condition
  A Horn formula is a conjunction of $(\bigwedge v_i = v_j) \Rightarrow v_k = v_l$,
  where v_i, v_j, v_k, and v_l are variables or constants
  Example: c=c $\Rightarrow z =$ d above (equivalent to $z =$ d)
- MVDs add equalities to local conditions
  Example: $x =$ a above

Literature

The section on incomplete information from *Foundations of Databases* by Abiteboul, Hull, and Vianu.
Data Cleaning: Problems and Current Approaches by Rahm and Do. (search with google scholar)

Inc0mplet$\substack{e \\ a}$ In?ormati*n