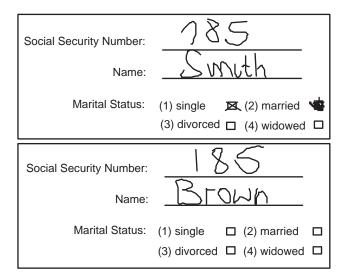
Incomplet^ea In?ormation

"Informationssysteme" SS 2007 Dan Olteanu, Lehrstuhl für Informationssysteme, Universität des Saarlandes

◆□▶ ◆□▶ ◆三▶ ◆三▶ ○三 のへぐ

Why Can Information Be Incomplete? (1)



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 |

◆□▶ ◆□▶ ◆ □▶ ★ □▶ = = - のへぐ

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 contradictions | 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:

- Data analysis (to detect the kind of occurring errors) data mining
- definition of transformation and mapping rules declarative SQL-based languages
- verification
- transformation
- 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

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

▲ロ ▶ ▲周 ▶ ▲ 国 ▶ ▲ 国 ▶ ● の Q @

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 (?).

- existing unknown values, e.g., DAO's phone or CEK's salary
- nonexisting values, e.g., LRA's phone
- Ino information is known about, e.g., LRA's room number

| Persons | | Salary | Room | Phone |
|---------|-------------------|------------|------|-------|
| | DAO LRA CEK | 40K 10K | 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 | <u>400K</u> | 226 | 57328 |
| | | | | |
| | | | | |
| Persons | Name | Salary | Room | Phone |
| Persons | Name DAO | Salary 40K | Room 228 | Phone <u>57332</u> |
| Persons | | 5 | | |
| Persons | DAO | 40K | 228 | |

.. and so on.

There is an infinite amount of possible worlds!!!

Represent intensionally the set of possible worlds

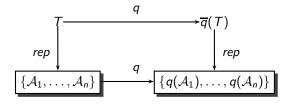
Representation Systems for Incomplete Information

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



< ロ > < 同 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ >

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

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

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

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

◆□▶ ◆□▶ ◆□▶ ◆□▶ ▲□ ◆ ○ ◆

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

 τ is a strong representation system for $\mathcal L$

Weak Representation Systems

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

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

 $\bigcap \{q(J) \mid J \in \mathcal{J}\}$ 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 au and q any query in \mathcal{L} , then

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

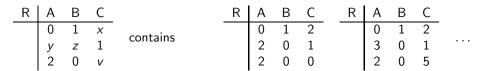
◆□▶ ◆□▶ ◆□▶ ◆□▶ ▲□ ◆ ○ ◆

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)

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



◆□▶ ◆□▶ ◆□▶ ◆□▶ ▲□ ◆ ○ ◆

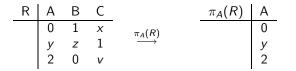
Querying Codd tables: Selection

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

Codd tables form no strong representation system for selection

▲ロ ▶ ▲周 ▶ ▲ 国 ▶ ▲ 国 ▶ ● の Q @

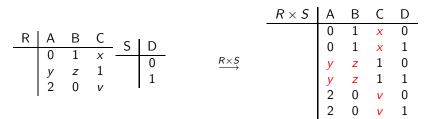
Querying Codd tables: Projection



Codd tables form a strong representation system for projection

◆□▶ ◆□▶ ◆三▶ ◆三▶ 三三 のへ⊙

Querying Codd tables: Product and Join



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 | А | R | C | | | | | | | $R \cup S$ | | | | |
|---|---|---|---|---|---|---|---|---|---|----------------------------|------------|---|---------------|---|--|
| - | | | | | - | ~ | | _ | ~ | | | 0 | 1 | x | |
| | | 0 | 1 | X | | S | Α | В | C | RUS | | | | | |
| | | у | 7 | 1 | | | 1 | 1 | v | $R \cup S \longrightarrow$ | | У | Z | T | |
| | | у | 2 | - | | | 1 | т | ~ | | | 2 | <i>z</i> 0 | V | |
| | | 2 | 0 | V | | | | | | | | 1 | 1 | | |
| | | • | | | | | | | | | | T | T | X | |

A variable can appear only once in a Codd table!

Codd tables form no strong representation system for union

◆□▶ ◆□▶ ◆ □▶ ★ □▶ = = - のへぐ

Querying Codd tables: Difference

Codd tables form no strong representation system for difference

◆□▶ ◆□▶ ◆ □▶ ★ □▶ = = - のへぐ

Certain Answers for Codd tables

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

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

- 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(σ_{A=3}(R), *T*) = Ø, 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 | А | В | С |
|---|---|---|---|
| | 0 | 1 | X |
| | у | Ζ | 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}_{θ} such that $\overline{q}_{\theta}(T) = \{t \mid t \in T, \forall \text{ valuations of vars in t } \mu : \theta(\mu(t))\}$
- 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) ≡_{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) ∈ sure(q ∘ q', T) but there are valuations such that (a', c) ∉ sure(q', W).

| | | | | $sure(q \circ q', T)$ | А | С |
|---|----|------------|----|-----------------------|---------|----|
| R | А | В | С | | a a' | С |
| | а | x x' | С | | a' | С |
| | a' | <i>x</i> ′ | c' | | a a' | c' |
| | - | | | | a' | c' |

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 } | | |
| | $\{ 185, 186 \}$ | Brown | $\{ 1, 2, 3, 4 \}$ |

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

| С | SSN | Name | MS | | С | SSN | Name | MS |
|------------|-----|-------|----|---|---|-----|-------|----|
| | 185 | Smith | 1 | _ | | 185 | Smith | 1 |
| | 185 | Brown | 1 | | | 185 | Brown | |
| C | SSN | Name | MS | | C | SSN | Name | MS |
| | | Smith | 1 | | | 185 | Smith | 1 |
| | 185 | Brown | 3 | | | 185 | Brown | 4 |
| and so on. | | | | | | | | |

(Naive) v-tables

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

| R | А | В | С |
|---|---|---|---|
| | 0 | 1 | X |
| | x | Ζ | 1 |
| | 2 | 0 | V |

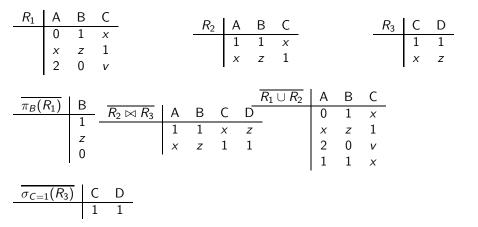
v-tables form a weak representation system for positive relational algebra

◆□▶ ◆□▶ ◆□▶ ◆□▶ ▲□ ◆ ○ ◆

Proof Idea

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

Querying v-tables



◆ロ▶ ◆母▶ ◆臣▶ ◆臣▶ 三臣 - のへで

(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.

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

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

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

A D M A

c-table Example (1)

| R_1 | Student | Course | |
|-------|-------------------------|---------|-------------------------------|
| | | | $x \neq math \land x \neq CS$ |
| | Sally Sally Sally | math | <i>z</i> = 0 |
| | Sally | CS | $z \neq 0$ |
| | Sally | X | |
| | Alice | bio | z = 0 |
| | Alice | math | $x = physics \land t = 0$ |
| | Alice | physics | $x = physics \land t \neq 0$ |

c-table Example (2)

| R_2 | Student | Course | |
|-------|---------|---------|-------------------------------|
| | | | true |
| | Sally | math | <i>z</i> = 0 |
| | Sally | CS | $z \neq 0$ |
| | Sally | x | $x \neq math \land x \neq CS$ |
| | Alice | bio | z = 0 |
| | Alice | math | $x = physics \land t = 0$ |
| | Alice | physics | $x = physics \land t \neq 0$ |

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

$$R_1 \neq R_2$$

◆□▶ ◆□▶ ◆ □▶ ★ □▶ = = - のへぐ

How Strong are c-tables? (1)

c-tables form a strong representation system 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)

| $\begin{array}{c cccc} T_1 & B & C \\ \hline & x & c \\ \hline & z & w \end{array} \xrightarrow{\begin{array}{c cccc}} T_2 & B & C \\ \hline & y & c & y=b \\ z & w \\ \end{array}$ | T ₃ A B a y |
|---|---|
| $\overline{\pi_B(T_2)}$ B | $\overline{T_1 \cup T_2}$ B C |
| $\begin{array}{c c} y & y = b \\ z \end{array}$ | $ \begin{array}{ccc} x & c \\ y & c & y = b \\ z & w \end{array} $ |
| $\overline{T_1 \bowtie T_3}$ A B C | |
| $\begin{array}{c cccc} & a & y & c & y = x \\ \hline \sigma_{B=b}(T_1 \bowtie T_3) & A & B & C \\ \hline & a & y & c & y = b \land y \end{array}$ | = x |
| $\overline{T_1 - T_2}$ B C | |
| ХС | $y \neq b \land x \neq z$ |
| | $y \neq b \land w \neq c$ $y = b \land x \neq b \land x \neq z$ |
| | $y = b \land x \neq b \land w \neq z$ $y = b \land x \neq b \land w \neq c$ |

How Strong are c-tables? (3)

| | c-tables can represent | c-tables can represent answers of transitive closure queries | | | | | | | | |
|--------------------|------------------------|--|---|---|-----------------------|--|--|--|--|--|
| | | | | | | | | | | |
| | | $\overline{tc}(T)$ | А | В | | | | | | |
| | | | а | b | x = b $x = d$ $x = b$ | | | | | |
| А | | | x | С | | | | | | |
| а | b | | с | d | | | | | | |
| a <i>x</i> c | С | | а | С | x = b | | | | | |
| с | d | | x | d | | | | | | |
| | | | с | С | x = d | | | | | |
| | | | а | d | x = b | | | | | |

▲□▶ ▲□▶ ▲□▶ ▲□▶ ▲□ ● ● ●

Overview of Representation Systems

| System | Is Weak For | Is Strong For | | |
|-------------------------|--------------------|---------------|--|--|
| Codd tables v-tables | PS | Р | | |
| v-tables | PS ⁺ UJ | PU | | |
| c-tables | PSUJD | PSUJD | | |

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

◆□▶ ◆□▶ ◆ □▶ ★ □▶ = = - のへぐ

Decision Problems for Representation Systems

◆□ > ◆□ > ◆ Ξ > ◆ Ξ > → Ξ = ∽ ۹ < ↔

Decision Problems

Representation system \mathcal{W} , instance $I = (R^{I})$, tuple t Input $\exists \mathcal{A} \in rep(\mathcal{W}) : t \in R^{\mathcal{A}}$ Problems Tuple Possibility: $\forall \mathcal{A} \in rep(\mathcal{W}) : t \in R^{\mathcal{A}}$ Tuple Certainty: $\exists \mathcal{A} \in rep(\mathcal{W}) : R' = R^{\mathcal{A}}$ Instance Possibility: $\forall \mathcal{A} \in rep(\mathcal{W}) : R' = R^{\mathcal{A}}$ Instance Certainty: $\exists \mathcal{A} \in rep(\mathcal{W}) : t \in Q(\mathcal{A})$ Tuple *Q*-Possibility (query Q fixed): Tuple Q-Certainty (query Q fixed): $\forall \mathcal{A} \in rep(\mathcal{W}) : t \in Q(\mathcal{A})$ $\exists \mathcal{A} \in rep(\mathcal{W}) : R^{I} = Q(\mathcal{A})$ Instance Q-Possibility (query Q fixed): $\forall \mathcal{A} \in rep(\mathcal{W}) : R' = Q(\mathcal{A})$ Instance Q-Certainty (query Q fixed):

◆□▶ ◆□▶ ◆□▶ ◆□▶ ▲□ ◆ ○ ◆

Decisions for c-tables

| R_1 | Student | Course | |
|-------|----------------------------------|---------|-------------------------------|
| | | | $x \neq math \land x \neq CS$ |
| | Sally Sally Sally Alice | math | <i>z</i> = 0 |
| | Sally | CS | $z \neq 0$ |
| | Sally | X | |
| | Alice | bio | z = 0 |
| | Alice | math | $x = physics \land t = 0$ |
| | Alice | physics | $x = physics \land t \neq 0$ |

- 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?
 Ø, {(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 $\forall x, y, z, y_1, z_1(R(x, y, z) \land R(x, y_1, z_1) \Rightarrow y = y_1)$.

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

Chasing Dependencies on World-Sets

Dependencies can be used to

- eliminate inconsistent worlds. This leads to less worlds thus more information.
- 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

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

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

Chasing Dependencies on v-tables

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

| R | А | В | С | | R' | А | В | С |
|---|---|---|---|--------------------------------|----|---|---|---|
| | 0 | 1 | X | Chase $B \rightarrow C$ on R | | 0 | 1 | 2 |
| | x | Ζ | 1 | Chase $B \rightarrow C$ on R | | 2 | Ζ | 1 |
| | 2 | 1 | 2 | | | 2 | 1 | 2 |

◆□▶ ◆□▶ ◆ □▶ ★ □▶ = = - のへぐ

Chasing Dependencies on c-tables

| $\Sigma = -$ | { <i>A</i> → | <i>→ B</i> , | $C \rightarrow$ | • D}, | c-table | es T_1 | and | T_2 . | The | n, | $chase_{\Sigma}(rep(T_1)) =$ | $= rep(T_2)$. |
|--------------|--------------|--------------|-----------------|-------|---------|----------|-----|---------|-----|----|------------------------------|----------------|
| | | | | | | T_2 | А | В | С | D | | |
| T_1 | | | | | | | | | | | $c=c \Rightarrow z = d$ | |
| | a X a | b | С | d | - | | а | b | С | d | | |
| | x | е | у | g | | | x | е | У | g | | |
| | а | b | С | Ζ | | | а | b | У | g | x = a x = a | |
| | - | | | | | | а | е | С | Ζ | 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)

Incomplet a In?ormati*n