Ontology and Database Systems: Foundations of Database Systems

Part 4: Incomplete Databases

Werner Nutt

Faculty of Computer Science
Master of Science in Computer Science

A.Y. 2014/2015

Freie Universität Bozen
Libera Università di Bolzano
Free University of Bozen · Bolzano

Incomplete Information

Schema

`Person(fname, surname, city, street)`

`City(cname, population)`

We know

- Mair lives in Bozen *(but we don't know first name and street)*
- Carlo Rossi lives in Bozen *(but we don't know the street)*
- Mair and Carlo Rossi live in the same street *(but we don't know which)*
- Maria Pichler lives in Brixen
- Bozen has a population of 100,500
- Brixen has a population $<$ 100,000 *(but we don't known the number)*

Queries

1. Return first name and surname of people living in Bozen!
2. Return the surnames of people living in Bozen!
3. Who (surname) is living in the same street as Mair?
4. Which people are living in a city with less than 100,000 inhabitants?

Incomplete Information: Questions

How can we represent this info?

- What does the representation look like?
- What is its meaning?
- What answers should the queries return?
- How can we define the semantics of queries?

Modeling Incomplete Information: SQL Nulls

Person

fname	surname	city	street
NULL	Mair	Bozen	NULL
Carlo	Rossi	Bozen	NULL
Maria	Pichler	Brixen	NULL

City

cname	population
Bozen	100,500
Brixen	NULL

Intuitive meaning of NULL: one of

(i) does not exist

(ii) exists, but is unknown

(iii) unknown whether (i) or (ii)

SQL Nulls: Formal Semantics

- **dom** (or, equivalenty, every type) is extended by a new value: `NULL`
- built-in predicates are evaluated according to a 3-valued logic with truth values *false* $<$ *unknown* $<$ *true*
- atoms with `NULL` evaluate to *unknown*
- Boolean operations:
  - AND/OR correspond to min/max on truth values
  - NOT extends the classical definition by NOT(*unknown*) $=$ *unknown*
- additional operation $\texttt{ISNULL}(\cdot)$ with $\texttt{ISNULL}(v) =$ *true* iff v is `NULL`
- a query returns those tuples for which query conditions evaluate to *true*

SQL Nulls: Example Queries

- Query 1 returns (`NULL`, Mair) and (Carlo, Rossi)
- Query 2 returns Mair and Rossi
- Queries 3 and 4 return nothing

Representation Systems [Imieliński/Lipski, 1984]

Distinguish between

- **semantic instances**, which are the ones we know
- **syntactic instances**, which contain tuples with variables (written $\perp_1$, $\perp_2, \ldots$)

A syntactic instance represents many semantic instances

Syntactic instances are called **multi-tables** (i.e., several tables).

There are three kinds of (multi-)tables:

Codd Tables: a variable occurs no more than once

Naive or Variable Tables: a variable can occur several times

Conditional Tables: variable table where each tuple $\bar{t}$ is tagged with a boolean combination *cond*$(\bar{t})$ of built-in atoms

Short names: table, v-table, c-table

Semantics of Tables

Let $\mathbf{T}$ be a multi-table with variables $\mathit{var}(\mathbf{T})$.
For an assignement $\alpha\colon \mathit{var}(\mathbf{T}) \to \mathbf{dom}$ we define

$$\alpha\mathbf{T} = \{\alpha\,\bar{t} \mid \bar{t} \in \mathbf{T},\ \alpha \models \mathit{cond}(\bar{t}\,)\}$$

Then $\mathbf{T}$ **represents** the infinite sets of instances

$$\mathit{rep}(\mathbf{T}) = \{\alpha\mathbf{T} \mid \alpha\colon \mathit{var}(\mathbf{T}) \to \mathbf{dom}\}$$

$$\mathit{Rep}(\mathbf{T}) = \{\mathbf{J} \mid \mathbf{I} \subseteq \mathbf{J} \text{ for some } \mathbf{I} \in \mathit{rep}(\mathbf{T})\}$$

where

$\mathit{rep}(\mathbf{T})$ is the *closed-world* interpretation of $\mathbf{T}$

$\mathit{Rep}(\mathbf{T})$ is the *open-world* interpretation of $\mathbf{T}$

(Many results hold for both, the closed-world and the open-world interpretation. We assume open-world interpretation if not said otherwise.)

Certain and Possible Answers

Given $\mathbf{T}$ and a query Q, the tuple $\bar{c}$ is

- a **certain answer** (for Q over $\mathbf{T}$) if
  $\bar{c}$ is returned by Q over **all** instances represented by $\mathbf{T}$
- a **possible answer** if
  $\bar{c}$ is returned by Q over **some** instance represented by $\mathbf{T}$

We denote the set of all certain answers as $\mathit{cert}_{\mathbf{T}}(Q)$.

We have

$$\mathit{cert}_{\mathbf{T}}(Q) \quad = \bigcap_{\mathbf{J} \in \mathit{Rep}(\mathbf{T})} Q(\mathbf{J})$$

Modeling Incomplete Information: Codd-Tables

Person

fname	surname	city	street
$\perp_1$	Mair	Bozen	$\perp_2$
Carlo	Rossi	Bozen	$\perp_3$
Maria	Pichler	Brixen	$\perp_4$

City

cname	population
Bozen	100,500
Brixen	$\perp_5$

Certain answers for our example queries:

- Query 1 returns (Carlo, Rossi)
- Query 2 returns Mair and Rossi
- Query 3 returns **Mair**
- Query 4 returns nothing

Modeling Incomplete Information: v-Tables

Person

fname	surname	city	street
$\bot_1$	Mair	Bozen	$\bot_2$
Carlo	Rossi	Bozen	$\bot_2$
Maria	Pichler	Brixen	$\bot_4$

City

cname	population
Bozen	100,500
Brixen	$\bot_5$

Certain answers for our example queries:

- Query 1 returns (Carlo, Rossi)
- Query 2 returns Mair and Rossi
- Query 3 returns Mair and **Rossi**
- Query 4 returns nothing

Modeling Incomplete Information: c-Tables

Person

fname	surname	city	street
$\perp_1$	Mair	Bozen	$\perp_2$
Carlo	Rossi	Bozen	$\perp_2$
Maria	Pichler	Brixen	$\perp_4$

City

cname	population	cond
Bozen	100,500	*true*
Brixen	$\perp_5$	$\perp_5 < 100,000$

Certain answers for our example queries:

- Query 1 returns (Carlo, Rossi)
- Query 2 returns Mair and Rossi
- Query 3 returns Mair and Rossi
- Query 4 returns **Pichler**

Strong Representation Systems

Definition

Let Q be a query and $\mathbf{T}$ be a table. Then

$$Q(\mathbf{T}) := \{Q(\mathbf{I}) \mid \mathbf{I} \in \mathit{rep}(\mathbf{T})\}$$

That is, $Q(\mathbf{T})$ contains the relation instances obtained by applying Q individually to each instance represented by $\mathbf{T}$.

Note: $Q(\mathbf{T})$ is a set of sets of tuples, not a set of tuples!

Strong Representation Systems (cont)

Theorem (Imieliński/Lipski)

For every relational algebra query Q and every c-table $\mathbf{T}$ one can compute a c-table $\widetilde{\mathbf{T}}$ such that

$$rep(\widetilde{\mathbf{T}}) = Q(\mathbf{T})$$

That is,

- $\widetilde{\mathbf{T}}$ can be considered the answer of Q over $\mathbf{T}$
- the result of querying a c-table can be represented by a c-table
  $\rightsquigarrow$ c-tables are a **strong representation system**

The downside:

- handling of c-tables is intractable:
  the membership problem "$\mathbf{I} \in rep(\mathbf{T})$"? is NP-hard
- the c-tables $\widetilde{\mathbf{T}}$ may be very large

Weak Representation Systems: Motivation

Let $\mathbf{T}_{\mathrm{v}}$ be our example v-table and consider

$$Q_0 = \pi_{\texttt{fname},\texttt{sname}}(\sigma_{\texttt{city}=\text{'Bozen'}}(\texttt{Person})),$$
$$Q_1 = \pi_{\texttt{sname}}(\sigma_{\texttt{city}=\text{'Bozen'}}(\texttt{Person}))$$

Then: $\mathit{cert}_{\mathbf{T}_{\mathrm{v}}}(Q_0) = \{(\text{Carlo, Rossi})\}$ and

$\mathit{cert}_{\mathbf{T}_{\mathrm{v}}}(Q_1) = \{(\text{Mair}), (\text{Rossi})\}$

Observation: $Q_0 = \pi_{\texttt{sname}}(Q_1)$,

but $\mathit{cert}_{\mathbf{T}_{\mathrm{v}}}(Q_0)$ cannot be computed from $\mathit{cert}_{\mathbf{T}_{\mathrm{v}}}(Q_1)$

Compositionality is violated! Better keep the nulls!

Idea: Try to keep enough information for representing certain answers!

Incomplete Databases: Definition

Definition (Incomplete Database)

An **incomplete database** is a set of instances ($\mathcal{I}$, $\mathcal{J}$).

For a query Q and an incomplete db $\mathcal{I}$, the set of certain answers for Q over $\mathcal{I}$ is

$$\mathit{cert}_{\mathcal{I}}(Q) := \bigcap_{\mathbf{I} \in \mathcal{I}} Q(\mathbf{I}).$$

Weak Representation Systems

Let $\mathcal{L}$ be a query language
(e.g., conjunctive queries, positive queries, positive relational algebra)

Definition ($\mathcal{L}$-Equivalence)

Two incomplete databases $\mathcal{I}$, $\mathcal{J}$ are $\mathcal{L}$-equivalent, denoted $\mathcal{I} \equiv_{\mathcal{L}} \mathcal{J}$, if for each $Q \in \mathcal{L}$ we have

$$\mathit{cert}_{\mathcal{I}}(Q) = \mathit{cert}_{\mathcal{J}}(Q)$$

That is, $\mathcal{L}$-equivalent incomplete dbs give rise to the same certain answers for all queries in $\mathcal{L}$.

Goal: For Q and $\mathbf{T}$, find a $\mathbf{T}'$ such that $\mathbf{T}'$ is $\mathcal{L}$-equivalent to $Q(\mathit{Rep}(\mathbf{T}))$, for a suitable $\mathcal{L}$

Weak Representation Systems (cntd)

$\mathcal{L}^{+}_{\text{calc}}$ language of positive relational calculus queries

Theorem (Imielinski/Lipski)

For every positive query Q and v-table $\mathbf{T}$, one can compute a v-table $\mathbf{T}'$ such that

$$\mathit{Rep}(\mathbf{T}') \equiv_{\mathcal{L}^{+}_{\text{calc}}} Q(\mathit{Rep}(\mathbf{T}))$$

Proof.

Apply Q to $\mathbf{T}$, treating variables like constants. □

That is, $\mathbf{T}'$

- contains enough information to compute certain answers to positive queries on $Q(\mathit{Rep}(\mathbf{T}))$
- can be considered the answer of Q over $\mathbf{T}$, in the context of positive queries