Scalable Uncertainty Management

02 – Incomplete Databases

Rainer Gemulla

April 27, 2012

Overview

In this lecture

- Refresh relational algebra
- What is an incomplete database?
- How can incomplete information be represented?
- How expressive are these representations?
- How to query incomplete databases?
- How to query their representations?

Not in this lecture

- Complexity
- Efficiency
- Applications

Outline

Notation

- Set of *attributes* $\mathscr{A}$ (countably infinite, totally ordered)
- *Domain* $\mathscr{D}$ of values for the attributes (countably infinite)
- Elements of $\mathscr{D}$ are called *constants*
- Per-attribute domain denoted $\mathrm{dom}(A)$
- Set of *relation names* $\mathscr{R}$, each associated with a finite set of attributes $\alpha(R) \subset \mathscr{A}$ (countably infinite names per finite set of attributes)
- A *schema* is a finite set of attributes (symbols U, W, V)
- A *relation schema* is a relation name (symbols R, S)
- A *database schema* is a nonempty finite set of relation names

Example

R

	A	B	C
t_1:	a_1	b_2	c_1
t_2:	a_2	b_1	c_1

- $\mathscr{A} = \{ A, B, C, D, \ldots \} = ABCD \ldots$
- $\mathscr{D} = \{ a_1, b_1, c_1, a_2, \ldots \}$
- $\mathrm{dom}(A) = \{ a_1, a_2, \ldots \}$
- $\mathscr{R} = \{ R, S, \ldots \}$
- $\alpha(R) = ABC$; write $R[ABC]$

The Named Perspective

- Let $U \subset \mathscr{A}$ be a schema
- *Tuple* t over U is a function $t : U \to \mathscr{D}$ (also called *U-tuple*)
- $\alpha(t)$ denotes the schema of t
- *Value* of attribute $A \in U$ of U-tuple t is denoted $t(A)$ or $t.A$
- *Restriction* of U-tuple t to values in $V \subseteq U$ is denoted $t[V]$
- *Relation instance* $I(R)$ of R is a finite set of tuples over $\alpha(R)$
- *Database instance* $\mathbf{I}$ of database schema $\mathbf{R}$ maps each relation name in $R \in \mathbf{R}$ to a relation instance $\mathbf{I}(R)$

Example

R

	A	B	C
t_1:	a_1	b_2	c_1
t_2:	a_2	b_1	c_1

- t_1 is a tuple over ABC
- $t_1 = \langle A\colon a_1, B\colon b_2, C\colon c_1 \rangle = a_1 b_2 c_1$
- $\alpha(t_1) = ABC$
- $t_1(A) = t_1.A = a_1$
- $t_1[AB] = a_1 b_2$ is a tuple over AB
- $I(R) = \{ t_1, t_2 \} = \{ a_1 b_2 c_1, a_2 b_1 c_1 \}$ is relation instance over ABC

The Unnamed Perspective

- *Tuple* t is an ordered n-tuple ($n \geq 0$) of constants, i.e., $t \in \mathscr{D}^n$
- *Value* of i-th coordinate denoted $t(i)$
- Natural correspondence to named perspective
  - n-tuples can be viewed as functions with domain $\{1, \ldots, n\}$
  - U-tuples can be viewed as $|U|$-tuples by using total order of attributes

Example

R

t_1:	a_1	b_2	c_1
t_2:	a_2	b_1	c_1

- $t_1 = \langle a_1, b_2, c_1 \rangle = a_1 b_2 c_1$
- $t_1(1) = a_1$

For now, we will mostly use the named perspective.

Relational algebra (1)

- Relation name R
- Single-tuple, single-attribute *constant relations* (`VALUES` clause)

$$\{\,\langle A\colon a\rangle\,\}$$

  for $A \in \mathscr{A}$, $a \in \mathrm{dom}(A)$
- *Selection* σ (`WHERE` clause)

$$\sigma_{A=a}(I) = \{\, t \in I \mid t.A = a \,\}$$
$$\sigma_{A=B}(I) = \{\, t \in I \mid t.A = t.B \,\}$$

  for $A, B \in \alpha(I)$ and $a \in \mathrm{dom}(A)$.

Example

R

	A	B	C
t_1:	a_1	b_2	c_1
t_2:	a_2	b_1	c_2
t_3:	a_1	b_1	c_1

$\{\,\langle A\colon a\rangle\,\}$

A
a

$\sigma_{A=a_1}(R)$

A	B	C
a_1	b_2	c_1
a_1	b_1	c_1

$\sigma_{A=a_3}(R)$

A	B	C

Relational algebra (2)

- *Projection* π (`SELECT DISTINCT` clause)

  $$\pi_U(I) = \{\, t[U] \mid t \in I \,\}$$

  for $U \subseteq \alpha(R)$

- *Natural Join* $\bowtie$ (`FROM clause`)

  $$I \bowtie J = \{\, t \text{ over } U \cup V \mid t[U] \in I \wedge t[V] \in J \,\},$$

  where $U = \alpha(I)$, $V = \alpha(J)$

Example

R

	A	B	C
t_1:	a_1	b_2	c_1
t_2:	a_2	b_1	c_2
t_3:	a_1	b_1	c_1

S

	A	D
t_4:	a_1	d_1
t_5:	a_3	d_2
t_6:	a_1	d_3

$\pi_{AC}(R)$

A	C
a_1	c_1
a_2	c_2

$R \bowtie S$

A	B	C	D
a_1	b_2	c_1	d_1
a_1	b_2	c_1	d_3
a_1	b_1	c_1	d_1
a_1	b_1	c_1	d_3

Relational algebra (3)

- *Renaming* of attributes ρ (AS clause)

$$\rho_{A_1 \dots A_n \to B_1 \dots B_n}(I) = \{\, t \text{ over } V \mid (\exists u \in I)(\forall i \in [1,n])\, u.A_i = t.B_i \,\},$$

where $\alpha(I) = \{\, A_1, \dots, A_n \,\}$, $V = \{\, B_1, \dots, B_n \,\}$

- Short notation: only list attributes being renamed

Example

R

	A	B
t_1:	a_1	b_2
t_2:	a_2	b_1
t_3:	a_1	b_1

$\rho_{AB \to CD}(R)$

C	D
a_1	b_2
a_2	b_1
a_1	b_1

$\rho_{AB \to BA}(R)$

B	A
a_1	b_2
a_2	b_1
a_1	b_1

$R \bowtie \rho_{B \to C}(R)$

A	B	C
a_1	b_2	b_2
a_1	b_2	b_1
a_2	b_1	b_1
a_1	b_1	b_1
a_1	b_1	b_2

Relational algebra (4)

- *Union* $\cup$ (`UNION` clause)

  $$I \cup J = \{\, t \mid t \in I \lor t \in J \,\}$$

  for $\alpha(I) = \alpha(J)$
- *Difference* $-$ (`EXCEPT` clause)

  $$I - J = \{\, t \mid t \in I \land t \notin J \,\}$$

  for $\alpha(I) = \alpha(J)$

Example

R

	A	B
t_1:	a_1	b_2
t_2:	a_2	b_1
t_3:	a_1	b_1

S

	A	B
t_4:	a_1	b_2
t_5:	a_2	b_1
t_6:	a_3	b_2

$R \cup S$

A	B
a_1	b_2
a_2	b_1
a_1	b_1
a_3	b_2

$R - S$

A	B
a_1	b_1

$\mathscr{L}$-expression

Definition

Let $\mathscr{L} \subseteq$ SPJRUD be an algebra. An $\mathscr{L}$*-expression* is any well-formed relational algebra expression composed of only relation names, constant relations, and the operations in $\mathscr{L}$. Algebra $\mathscr{L}$ is *positive* if it does not contain the difference operator.

Example

- $\pi_A(\pi_{AB}(R))$ is a P-expression but not an S-expression
- $\sigma_{A=a}(R)$ is both an S-expression and a PS-expression, but not a P-expression
- R is an $\emptyset$-expression
- All of the above expressions are positive, but $R - S$ is not

Generalized Selection

- Relational algebra
  - $\sigma_{A=a}(R)$ for $A \in \alpha(R)$ and $a \in \mathrm{dom}(A)$
  - $\sigma_{A=B}(R)$ for $A, B \in \alpha(R)$
  - $A = a$ and $A = B$ are called *predicates*
- Generalized selection operators extend the class of predicates
- Positive conjunction

$$\sigma_{P_1 \wedge P_2}(R) = \sigma_{P_1}(\sigma_{P_2}(R))$$

- Positive disjunction (S^+)

$$\sigma_{P_1 \vee P_2}(R) = \sigma_{P_1}(R) \cup \sigma_{P_2}(R)$$

- Negation (S^-, not positive)

$$\sigma_{\neg P}(R) = R - \sigma_P(R)$$

- Note: Union and difference can simulate generalized selection but not vice versa! $\rightarrow$ S^+ and S^- variants of S

Outline

Examples of incomplete information

Certain data	Uncertain data	
Paul owns a car.	Paul may own a car.	Tuple-level uncertainty

Name	Object
Paul	Car

Name	Object
Paul	Car

or

Name	Object

Bob works for Yahoo.

Name	Company
Bob	Yahoo

Bob works for either Yahoo or Microsoft.

Name	Company
Bob	Yahoo

or

Name	Company
Bob	Microsoft

Attribute-level uncertainty

Mary sighted a finch.
Paul sighted a finch.

Name	Bird
Mary	Finch
Paul	Finch

Mary sighted a finch or a sparrow.
Paul sighted what Mary sighted.

Name	Bird
Mary	Finch
Paul	Finch

or

Name	Bird
Mary	Sparrow
Paul	Sparrow

Correlations

Paul's favorite number is 17.

Name	Num
Paul	17

Paul has a favorite number, but it is unknown.

Name	Num
Paul	1

or

Name	Num
Paul	2

or . . .

Infinity

We need a precise way to *model* and *represent* incomplete information.

Examples of incomplete databases

Certain data	Uncertain data	
Paul owns a car.	Paul may own a car.	Tuple-level uncertainty
Bob works for Yahoo.	Bob works for either Yahoo or Microsoft.	Attribute-level uncertainty
Mary sighted a finch. Paul sighted a finch.	Mary sighted a finch or a sparrow. Paul sighted what Mary sighted.	Correlations
Paul's favorite number is 17.	Paul has a favorite number, but it is unknown.	Infinity

Tuple-level uncertainty

Certain data:

Name	Object
Paul	Car

Uncertain data: {

Name	Object
Paul	Car

,

Name	Object

}

Attribute-level uncertainty

Certain data:

Name	Company
Bob	Microsoft

Uncertain data: {

Name	Company
Bob	Yahoo

,

Name	Company
Bob	Microsoft

}

Correlations

Certain data:

Name	Bird
Mary	Finch
Paul	Finch

Uncertain data: {

Name	Bird
Mary	Finch
Paul	Finch

,

Name	Bird
Mary	Sparrow
Paul	Sparrow

}

Infinity

Certain data:

Name	Num
Paul	17

Uncertain data: {

Name	Num
Paul	1

,

Name	Num
Paul	2

, . . . }

An incomplete database is a set of "possible worlds" (i.e., DB instances).

Incomplete database

$\mathscr{N}_U = \{\, I \mid I \text{ is a (finite) relation instance over schema } U \,\}$

Definition

- An *incomplete relation* (*i-relation*) $\mathcal{I}$ over U is a set of possible relation instances over U, i.e., $\mathcal{I} \subseteq \mathscr{N}_U$.
- An *incomplete database* (*i-database*) of a database schema $\mathbf{R}$ maps each relation name $R \in \mathbf{R}$ to an incomplete relation over $\alpha(R)$.

- "Incomplete" refers to incomplete information
- Focus on one relation $\rightarrow$ use i-relation and i-database synonymously
- Usual relation instances: $\mathcal{I} = \{\, I \,\}$
- *No-information* or *zero-information database* over U: $\mathcal{I} = \mathscr{N}_U$
- Incomplete databases can be *infinite* even though every relation instance is finite; e.g., $\{\, \boxed{a_1}, \boxed{a_2}, \boxed{a_3}, \dots \,\}$
- $\mathscr{N}_U$ is (countably) infinite
- Set of all incomplete relations is uncountable

Representation system

- Incomplete databases are in general infinite
- Even if finite, they can be very large

$\rightarrow$ Need compact representation!

Definition

A *representation system* consists of a set (a "language") $\mathscr{T}$ whose elements we call *tables*, and a function Mod that associates to each table $T \in \mathscr{T}$ an incomplete database $\mathrm{Mod}(T)$.

- Again, we'll assume a single relation
  (reformulation for multiple relations possible)
- $\mathrm{Mod}(T)$ can be thought of as the set of database instances consistent with T (called the *possible worlds*)
- T can be viewed as logical assertion; $\mathrm{Mod}(T)$ are *models* of T

Codd tables

- Missing values are indicated by a single, untyped *null value* @
- Each occurrence of @ stands for a value of the attribute's domain
- Different occurrences may or may not refer to the same value

Example

SUPPLIER	LOCATION	PRODUCT	QUANTITY
Smith	London	Nails	@
Brown	@	Bolts	@
Jones	@	Nuts	40,000

Definition

An *@-tuple* on U is an extended tuple in which each attribute $A \in U$ takes values in $\mathrm{dom}(A) \cup \{ @ \}$. A *Codd table* is a finite set of @-tuples.

Models of Codd tables (1)

Definition

Under the *closed world interpretation*, a Codd table represents the set of relations obtained by replacing @-symbols by valid values.

Example

Suppose $\mathrm{dom}(A) = \{ a_1, a_2 \}$ and $\mathrm{dom}(B) = \{ b_1, b_2 \}$.

$$\mathrm{Mod}\left(\begin{array}{|c|c|}\hline a_1 & @ \\ @ & b_2 \\ \hline\end{array}\right) = \left\{ \begin{array}{|c|c|}\hline a_1 & b_1 \\ a_1 & b_2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_1 \\ a_2 & b_2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_2 \\ a_2 & b_2 \\ \hline\end{array} \right\}$$

Let $R^* \in$ RHS of the example:

- There is no certain tuple, i.e., $\nexists t \forall R^*\ t \in R^*$
- The first column contains a_1, the second b_2
- R^* has at least one and at most 2 tuples
- $a_2 b_1$ is not in R^*
- ...

Negative information *can* be represented.

Models of Codd tables (2)

Definition

Under the *open world interpretation*, a Codd table represents the set of relations obtained by replacing @-symbols by valid values and adding arbitrarily many additional tuples.

Equivalently, this means $S \in \mathrm{MOD}(T) \iff (\exists R)\, R \in \mathrm{Mod}(T) \land S \supseteq R$.

Example

$$\mathrm{MOD}\left(\overset{T}{\boxed{\begin{array}{c|c} a_1 & @ \\ @ & b_2 \end{array}}}\right) = \left\{ \begin{array}{l} \boxed{\begin{array}{c|c} a_1 & \color{red}{b_2} \end{array}}, \boxed{\begin{array}{c|c} a_1 & \color{red}{b_1} \\ \color{red}{a_1} & b_2 \end{array}}, \boxed{\begin{array}{c|c} a_1 & \color{red}{b_1} \\ \color{red}{a_2} & b_2 \end{array}}, \boxed{\begin{array}{c|c} a_1 & \color{red}{b_2} \\ \color{red}{a_2} & b_2 \end{array}}, \boxed{\begin{array}{c|c} a_1 & \color{red}{b_2} \\ a_2 & b_1 \end{array}}, \\ \boxed{\begin{array}{c|c} a_1 & \color{red}{b_1} \\ \color{red}{a_1} & b_2 \\ a_2 & b_1 \end{array}}, \boxed{\begin{array}{c|c} a_1 & \color{red}{b_1} \\ \color{red}{a_1} & b_2 \\ a_2 & b_2 \end{array}}, \boxed{\begin{array}{c|c} a_1 & \color{red}{b_1} \\ \color{red}{a_2} & b_2 \\ a_2 & b_1 \end{array}}, \boxed{\begin{array}{c|c} a_1 & \color{red}{b_2} \\ \color{red}{a_2} & b_2 \\ a_2 & b_1 \end{array}}, \boxed{\begin{array}{c|c} a_1 & \color{red}{b_1} \\ \color{red}{a_1} & b_2 \\ a_2 & b_1 \\ a_2 & b_2 \end{array}} \end{array} \right\}$$

Models of Codd tables (3)

Example

$$\mathrm{MOD}\left(\overset{T}{\begin{array}{|c|c|}\hline a_1 & @ \\ @ & b_2 \\ \hline\end{array}}\right) = \left\{ \begin{array}{l} \begin{array}{|c|c|}\hline a_1 & b_2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_1 \\ a_1 & b_2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_1 \\ a_2 & b_2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_2 \\ a_2 & b_2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_2 \\ a_2 & b_1 \\ \hline\end{array}, \\ \begin{array}{|c|c|}\hline a_1 & b_1 \\ a_1 & b_2 \\ a_2 & b_1 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_1 \\ a_1 & b_2 \\ a_2 & b_2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_1 \\ a_2 & b_2 \\ a_2 & b_1 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_2 \\ a_2 & b_2 \\ a_2 & b_1 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_1 \\ a_1 & b_2 \\ a_2 & b_1 \\ a_2 & b_2 \\ \hline\end{array} \end{array} \right\}$$

Let $R^* \in$ RHS of the example:

- There is no certain tuple, i.e., $\nexists t \forall R^*\ t \in R^*$
- The first column contains a_1, the second b_2
- R^* has at least one tuple
- Every tuple is possible, i.e., $\forall t \exists R^*\ t \in R^*$
- ...

Negative information *cannot* be represented.

v-Tables

- Missing values are indicated by *marked null values* or variables
- $V(A)$ = set of *variables* for attribute A (countably infinite)
- $V(A) \cap V(B) = \emptyset$ if $\mathrm{dom}(A) \neq \mathrm{dom}(B)$; otherwise $V(A) = V(B)$

Example

Course	Teacher	Weekday
Databases	x	Monday
Programming	y	Tuesday
Databases	x	Thursday
FORTRAN	Smith	z

Definition

A *v-tuple* on U is an extended tuple in which each attribute $A \in U$ takes values in $dom(A) \cup V(A)$. A *v-table* is a finite set of *v*-tuples.

Models of v-tables

Example

Suppose $\mathrm{dom}(A) = \{ a_1, a_2 \}$, $\mathrm{dom}(B) = \{ b_1, b_2 \}$, $\mathrm{dom}(C) = \{ c_1, c_2 \}$.

$$\mathrm{Mod}\left(\begin{array}{|c|c|}\hline a_1 & x \\ y & b_2 \\ \hline\end{array}\right) = \left\{\begin{array}{|c|c|}\hline a_1 & b_1 \\ a_1 & b_2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_1 \\ a_2 & b_2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_2 \\ a_2 & b_2 \\ \hline\end{array}\right\}$$

$$\mathrm{Mod}\left(\begin{array}{|c|c|}\hline c_1 & z \\ z & c_2 \\ \hline\end{array}\right) = \left\{\begin{array}{|c|c|}\hline c_1 & c_1 \\ c_1 & c_2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline c_1 & c_2 \\ c_2 & c_2 \\ \hline\end{array}\right\}$$

$$\mathrm{Mod}\left(\begin{array}{|c|c|}\hline z_1 & z_2 \\ \hline\end{array}\right) = \left\{\begin{array}{|c|c|}\hline c_1 & c_1 \\ \hline\end{array}, \begin{array}{|c|c|}\hline c_1 & c_2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline c_2 & c_1 \\ \hline\end{array}, \begin{array}{|c|c|}\hline c_2 & c_2 \\ \hline\end{array}\right\}$$

- $\mathrm{Var}(T) = \{ x \mid \text{variable } x \text{ occurs in T} \}$
- *Valuation* $v : \mathrm{Var}(T) \to \mathscr{D}$ assigns (valid) values to each variable
- $v(T)$ is the relation obtained by replacing all variables by their values
- $\mathrm{Mod}(T) = \{ v(T) \mid v \text{ is a valuation for } \mathrm{Var}(T) \}$

Codd tables $\equiv$ v-tables in which each variable occurs at most once.

v-Tables and view updates

v-tables appear naturally when updating relational views.

Example

SL

Supplier	Location
Smith	London
x	New York
y	Los Angeles

SP

Supplier	Product
Smith	Nails
x	Bolts
y	Nuts

$\pi_{\text{Location,Product}}(\text{SL} \bowtie \text{SP})$

Location	Product
London	Nails
New York	Bolts
Los Angeles	Nuts

c-Tables

- *c-tables* are v-tables with an additional *condition* column *con*, indicating a "tuple existence condition" $\rightarrow$ *conditional table*
- *Conditions* taken from a set $\mathscr{C}$ composed of
  - false, true
  - $x = a$ for $x \in V(A)$ and $a \in \mathrm{dom}(A)$ for some $A \in \mathscr{A}$
  - $x = y$ for $x, y \in V(A)$ for some $A \in \mathscr{A}$
  - negation $\neg$, disjunction $\vee$, conjunction $\wedge$
- *Positive conditions* do not contain negations (set $\mathscr{C}^+$)

Example

Supplier	Location	Product	con
x	London	Nails	$x = \text{Smith}$
Brown	New York	Nails	$x \neq \text{Smith}$

Definition

A *c-tuple* t on U is an extended tuple over $U \cup \{con\}$ such that $t[U]$ is a v-tuple and $t(con) \in \mathscr{C}$. A *c-table* is a finite set of c-tuples.

Models of c-Tables

Example

Suppose $dom(x) = dom(y) = \{1, 2\}$.

$$\text{Mod}\left(\begin{array}{|c|c|c|}\hline A & B & con \\ \hline a_1 & b_1 & x = 1 \\ a_2 & b_1 & x \neq 1 \\ a_3 & b_2 & y = 1 \wedge x \neq 1 \\ a_4 & b_2 & y \neq 1 \vee x = 1 \\ \hline \end{array}\right) = \left\{ \begin{array}{c} \text{x1y1} \\ \begin{array}{|c|c|}\hline a_1 & b_1 \\ a_4 & b_2 \\ \hline \end{array} \end{array}, \begin{array}{c} \text{x1y2} \\ \begin{array}{|c|c|}\hline a_1 & b_1 \\ a_4 & b_2 \\ \hline \end{array} \end{array}, \begin{array}{c} \text{x2y1} \\ \begin{array}{|c|c|}\hline a_2 & b_1 \\ a_3 & b_2 \\ \hline \end{array} \end{array}, \begin{array}{c} \text{x2y2} \\ \begin{array}{|c|c|}\hline a_2 & b_1 \\ a_4 & b_2 \\ \hline \end{array} \end{array} \right\}$$

$$= \left\{ \begin{array}{|c|c|}\hline a_1 & b_1 \\ a_4 & b_2 \\ \hline \end{array}, \begin{array}{|c|c|}\hline a_2 & b_1 \\ a_3 & b_2 \\ \hline \end{array}, \begin{array}{|c|c|}\hline a_2 & b_1 \\ a_4 & b_2 \\ \hline \end{array} \right\}$$

- Valuation check conditions: $v(T) = \{ v(t[U]) \mid v(t(con)) = \text{true} \}$
- $\text{Mod}(T) = \{ v(T) \mid v \text{ is a valuation for } \text{Var}(T) \}$

v-tables are equivalent to c-tables in which each condition equals true.

Finite representation systems

Definition

In a *finite-domain* Codd-table, v-table, or c-table T, each variable $x \in \mathrm{Var}(T)$ is associated with a finite domain $\mathrm{dom}(x)$.

- Important in practice
- Sometimes easier to study
- Basis for most probabilistic databases
- Incomplete database is finite
  (but attribute domain and no. variables still countably infinite)

Other finite representation systems

All of these models can be seen as special cases of finite-domain c-tables.

Example

In *?-tables*, tuples are marked with ? if they may not exist.

$$\mathrm{Mod}\left(\begin{array}{|c|c|}\hline a_1 & b_1 \\ a_1 & b_2 \\ \hline \end{array}\,?\right) = \left\{ \begin{array}{|c|c|}\hline a_1 & b_1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline a_1 & b_1 \\ a_1 & b_2 \\ \hline \end{array} \right\}$$

In *or-set tables*, $t.A$ takes values in a finite subset of $\mathrm{dom}(A)$.

$$\mathrm{Mod}\left(\begin{array}{|c|c|}\hline a_1 & b_2 \\ a_1 & b_1 \| b_2 \\ a_2 & b_1 \| b_2 \\ \hline \end{array}\right) = \left\{ \begin{array}{|c|c|}\hline a_1 & b_2 \\ a_1 & b_1 \\ a_2 & b_1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline a_1 & b_2 \\ a_2 & b_1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline a_1 & b_2 \\ a_1 & b_1 \\ a_2 & b_2 \\ \hline \end{array}, \begin{array}{|c|c|}\hline a_1 & b_2 \\ a_2 & b_2 \\ \hline \end{array} \right\}$$

Equivalent to finite-domain Codd tables.

In a *?-or-set table*, both are combined.

$$\mathrm{Mod}\left(\begin{array}{|c|c|}\hline a_1 & b_1 \\ a_2 & b_1 \| b_2 \\ \hline \end{array}\,?\right) = \left\{ \begin{array}{|c|c|}\hline a_1 & b_1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline a_1 & b_1 \\ a_2 & b_1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline a_1 & b_1 \\ a_2 & b_2 \\ \hline \end{array} \right\}$$

Outline

Possible answer set semantics

Definition

The *possible answer set* to a query q on an incomplete database $\mathcal{I}$ is the incomplete database $q(\mathcal{I}) = \{\, q(I) \mid I \in \mathcal{I} \,\}$.

Example

Let $q(R) = \sigma_{A=a_1}(R)$.

$$q\left(\left\{ \begin{array}{|c|c|}\hline a_1 & b_1 \\ a_1 & b_2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_1 \\ a_2 & b_1 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_1 \\ a_2 & b_2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_2 & b_1 \\ \hline\end{array} \right\}\right) = \left\{ \begin{array}{|c|c|}\hline a_1 & b_1 \\ a_1 & b_2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_1 \\ \hline\end{array}, \begin{array}{|c|c|}\hline & \\ \hline\end{array} \right\}$$

Can we compute the representation of the possible answer set to a query from the representation of an incomplete database?

Strong representation systems

Definition

- A representation system is *closed* under a query language if for any query q and any table T there is a table $\bar{q}(T)$ that represents $q(\mathrm{Mod}(T))$.
- If $\bar{q}(T)$ can always be computed from q and T, the representation system is called *strong* under the query language.

$$
\begin{tikzcd}
T \arrow[r, "\mathrm{Mod}"] \arrow[d, "\bar{q}"'] & \mathcal{I} \arrow[d, "q"] \\
\bar{q}(T) \arrow[r, "\mathrm{Mod}"] & q(\mathcal{I})
\end{tikzcd}
$$

Intuitively, this means that the query language is "fully supported" by the representation system: query answers can be both computed and represented.

Normalized c-tables

Definition

A c-table T on U is *normalized* if $t[U] \neq t'[U]$ for all pairs of distinct c-tuples $t, t' \in T$.

Example

Not normalized

a_1	b_1	$x = 1$
a_1	b_1	$x = 2$
a_2	b_2	true

Normalized

a_1	b_1	$x = 1 \vee x = 2$
a_2	b_2	true

To normalize a c-table, repeatedly apply rule 3 (next slide).

We'll assume normalized c-tables throughout.

Mod-equivalence

Definition

Two tables T and T' are Mod-*equivalent* (or just *equivalent*) if $\text{Mod}(T) = \text{Mod}(T')$. We write $T \equiv_{\text{Mod}} T'$.

Mod-equivalent transformations on c-table T on U:

1. Replace a condition by an equivalent condition;
   e.g., $(x = 1 \wedge y = 1) \vee (x \neq 1 \wedge y = 1)$ by $y = 1$
2. Remove tuples in which condition is unsatisfiable;
   e.g., $x = 1 \wedge x = 2$
3. Merge tuples $t_1, \ldots, t_k \in T$ with $t_1[U] = \cdots = t_k[U]$ into a new tuple t' s.t. $t'[U] = t_1[U]$ and $t'.con = t_1.con \vee \cdots \vee t_k.con$.

Mod-equivalent transformations can be used to simplify c-tables.

c-Tables are strong

Theorem

c-tables, finite-domain c-tables, and Boolean c-tables are strong under $\mathcal{RA}$.

Proof.

Given a $\mathcal{RA}$ query q, construct $\bar{q}$ by replacing in q the operators π, σ, $\bowtie$, $\cup$, and $-$ by the respective operators $\bar{\pi}, \bar{\sigma}, \bar{\bowtie}, \bar{\cup}, \bar{-}$ of the *c-table algebra*. Then $v(\bar{q}(T)) = q(v(T))$ for all valuations v for $\mathrm{Var}(T)$. $\square$

- We assume and produce normalized c-tables
- Boolean c-table: all variables are boolean
- $T(t)$ denotes $t.con$ if $t \in T$; false otherwise
- $T[]$ drops condition column of normalized c-table
- Relational algebra operations on $T[]$ treat variables as normal values

c-Projection

Definition

$$\bar{\pi}_U(T)[] = \pi_U(T[])$$

$$\bar{\pi}_U(T)(t) = \bigvee_{t' \in T \text{ s.t. } t'[U]=t} T(t')$$

Example

Sightings

Name	*Species*	*con*
Anna	Guan	$x = 1$
Anna	Humming bird	$x = 2$
Bob	y	$x = 3$
z	Guan	$x = 4$

$\bar{\pi}_{Name}$(Sightings)

Name	*con*
Anna	$x = 1 \vee x = 2$
Bob	$x = 3$
z	$x = 4$

c-Selection

Definition

$$\bar{\sigma}_P(T)[] = T[]$$
$$\bar{\sigma}_P(T)(t) = T(t) \wedge P(t),$$

where $P(t)$ replaces in P each occurence of an attribute A by $t.A$ and evaluates subexpressions of form $a = b$ (to false) and $a = a$ (to true).

Example

Sightings

N	S	con
A	G	$x = 1$
A	H	$x = 2$
B	y	$x = 3$
z	G	$x = 4$

$\bar{\sigma}_{Species=\text{Guan}}$(Sightings)

N	S	con
A	G	$x = 1 \wedge \text{true}$
A	H	$x = 2 \wedge \text{false}$
B	y	$x = 3 \wedge y = \text{G}$
z	G	$x = 4 \wedge \text{true}$

$\bar{\sigma}_{S=\text{G}}$(Sightings) (simpl.)

N	S	con
A	G	$x = 1$
B	y	$x = 3 \wedge y = \text{G}$
z	G	$x = 4$

c-Union

Definition

$$(T_1 \bar{\cup} T_2)[] = T_1[] \cup T_2[]$$
$$(T_1 \bar{\cup} T_2)(t) = T_1(t) \vee T_2(t)$$

Example

Sightings

N	con
A	$x = 1$
B	$x = 2$
C	$x = 3$

VIPs

N	con
B	$y = 1$
C	$y = 2$
z	$y = 3$

Sightings $\bar{\cup}$ VIPs

N	con
A	$x = 1 \vee \text{false}$
B	$x = 2 \vee y = 1$
C	$x = 3 \vee y = 2$
z	$\text{false} \vee y = 3$

S$\bar{\cup}$V (simplified)

N	con
A	$x = 1$
B	$x = 2 \vee y = 1$
C	$x = 3 \vee y = 2$
z	$y = 3$

c-Join (1)

Definition

Set $U_1 = \alpha(T_1)$, $U_2 = \alpha(T_2)$, and denote by $V = U_1 \cap U_2 = A_1 \ldots A_k$ the join attributes. Let $V' = A_1' \ldots A_k'$ be a fresh set of attributes (of matching domains). Set $T_2' = \rho_{V \to V'}(T_2)$ and $U_2' = \alpha(T_2')$.

$$(T_1 \bar{\bowtie}_{V \to V'} T_2)[] = T_1[] \bowtie T_2'[]$$

$$(T_1 \bar{\bowtie}_{V \to V'} T_2)(t) = T_1(t[U_1]) \wedge T_2'(t[U_2']) \bigwedge_{A \in V} t.A = t.A'$$

$$T_1 \bar{\bowtie} T_2 = \bar{\pi}_{U_1 \cup U_2}(T_1 \bar{\bowtie}_{V \to V'} T_2').$$

c-Join (2)

Example

Sightings

N	S	con
A	G	$x = 1$
A	H	$x = 2$
z_1	K	$x = 3$
z_2	L	$x = 4$

VIPs

N	con
A	$y = 1$
B	$y = 2$
z_1	$y = 3$

VIPs′

N'	con
A	$y = 1$
B	$y = 2$
z_1	$y = 3$

Sightings $\bar{\bowtie}_{N \rightarrow N'}$ VIPs

N	S	N'	con
A	G	A	$x = 1 \wedge y = 1 \wedge \text{true}$
A	H	A	$x = 2 \wedge y = 1 \wedge \text{true}$
z_1	K	A	$x = 3 \wedge y = 1 \wedge z_1 = \text{A}$
z_2	L	A	$x = 4 \wedge y = 1 \wedge z_2 = \text{A}$
A	G	B	$x = 1 \wedge y = 2 \wedge \text{false}$
A	H	B	$x = 2 \wedge y = 2 \wedge \text{false}$
z_1	K	B	$x = 3 \wedge y = 2 \wedge z_1 = \text{B}$
z_2	L	B	$x = 4 \wedge y = 2 \wedge z_2 = \text{B}$
A	G	z_1	$x = 1 \wedge y = 3 \wedge z_1 = \text{A}$
A	H	z_1	$x = 2 \wedge y = 3 \wedge z_1 = \text{A}$
z_1	K	z_1	$x = 3 \wedge y = 3 \wedge z_1 = z_1$
z_2	L	z_1	$x = 4 \wedge y = 3 \wedge z_2 = z_1$

c-Join (3)

Example (continued)

Sightings

N	S	con
A	G	$x1$
A	H	$x2$
z_1	K	$x3$
z_2	L	$x4$

VIPs

N	con
A	$y1$
B	$y2$
z_1	$y3$

VIPs′

N'	con
A	$y1$
B	$y2$
z_1	$y3$

Sightings$\bar{\bowtie}_{N \to N'}$VIPs (simplified)

N	S	N'	con
A	G	A	$x1y1$
A	H	A	$x2y1$
z_1	K	A	$x3y1 \wedge z_1 = \text{A}$
z_2	L	A	$x4y1 \wedge z_2 = \text{A}$
z_1	K	B	$x3y2 \wedge z_1 = \text{B}$
z_2	L	B	$x4y2 \wedge z_2 = \text{B}$
A	G	z_1	$x1y3 \wedge z_1 = \text{A}$
A	H	z_1	$x2y3 \wedge z_1 = \text{A}$
z_1	K	z_1	$x3y3$
z_2	L	z_1	$x4y3 \wedge z_2 = z_1$

Sightings$\bar{\bowtie}$VIPs (simplified)

N	S	con
A	G	$x1y1 \vee (x1y3 \wedge z_1 = \text{A})$
A	H	$x2y1 \vee (x2y3 \wedge z_1 = \text{A})$
z_1	K	$(x3y1 \wedge z_1 = \text{A}) \vee (x3y2 \wedge z_1 = \text{B}) \vee x3y3$
z_2	L	$(x4y1 \wedge z_2 = \text{A}) \vee (x4y2 \wedge z_2 = \text{B}) \vee (x4y3 \wedge z_2 = z_1)$

c-Difference

Definition (c-Table difference)

$$(T_1 \bar{-} \text{VIPs})[] = T_1[]$$

$$(T_1 \bar{-} \text{VIPs})(t) = T_1(t) \bigwedge_{t' \in \text{VIPs}} \neg(t = t' \wedge \text{VIPs}(t'))$$

Example

Sightings

A	con
A	$x1$
B	$x2$
C	$x3$

VIPs

A	con
B	$y1$
C	$y2$
z	$y3$

Sightings$\bar{-}$VIPs (simplified)

A	con
A	$x1 \wedge \neg(z = \text{A} \wedge y3)$
B	$x2 \wedge \neg y1 \wedge \neg(z = \text{B} \wedge y3)$
C	$x3 \wedge \neg y2 \wedge \neg(z = \text{C} \wedge y3)$

Sightings$\bar{-}$VIPs

A	con
A	$x1 \wedge \neg(\text{false} \wedge y1) \wedge \neg(\text{false} \wedge y2) \wedge \neg(z = \text{A} \wedge y3)$
B	$x2 \wedge \neg(\text{true} \wedge y1) \wedge \neg(\text{false} \wedge y2) \wedge \neg(z = \text{B} \wedge y3)$
C	$x3 \wedge \neg(\text{false} \wedge y1) \wedge \neg(\text{true} \wedge y2) \wedge \neg(z = \text{C} \wedge y3)$

Many representation systems are not closed

Theorem

Codd tables, v-tables, finite-domain Codd tables, finite-domain v-tables, ?-tables, or-set tables, and ?-or-set tables are not closed under $\mathcal{RA}$.

Proof.

By counterexample. Consider:

- Codd tables / v-tables (standard and finite-domain), or-set tables, ?-or-set tables:

$$\sigma_{A \neq B}\left(\begin{array}{|c|c|}\hline A & B \\ \hline x & y \\ \hline \end{array}\right) \qquad \text{where } \mathrm{dom}(x) = \mathrm{dom}(y) \text{ and } |\mathrm{dom}(x)| > 1.$$

- ?-tables:

$$\begin{array}{|c|c|}\hline A & B \\ \hline a_1 & b_1 \\ a_1 & b_2 \\ \hline \end{array} \bowtie \begin{array}{|c|}\hline A \\ \hline a_1 \\ \hline \end{array}?$$

We will see: these systems are still very useful!

□

Outline

Expressive power

Key question: How expressive is a given representation system?

Theorem

Neither Codd tables, v-tables, nor c-tables can represent all possible incomplete databases.

Proof.

Set of incomplete databases is uncountable, set of tables is countable. □

- E.g., zero-information database $\mathscr{N}_U$ cannot be represented with closed world assumption
- Need to study weaker forms of expressiveness
  1. $\mathcal{RA}$-completeness
  2. Finite completeness

$\mathcal{RA}$-definability (1)

- $\mathcal{Z}_V = \{ \{ t \} \mid \alpha(t) = V \}$
- $\mathcal{Z}_V$ is the minimal-information database for instances of cardinality 1

Example

Let $V = B_1 B_2$, where $\mathrm{dom}(B_1) = \mathrm{dom}(B_2) = \{ 1, 2, \ldots \}$.

$$\mathcal{Z}_V = \left\{ \begin{array}{|c|c|}\hline B_1 & B_2 \\ \hline 1 & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline B_1 & B_2 \\ \hline 1 & 2 \\ \hline \end{array}, \begin{array}{|c|c|}\hline B_1 & B_2 \\ \hline 2 & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline B_1 & B_2 \\ \hline 2 & 2 \\ \hline \end{array}, \ldots \right\}$$

Definition

An incomplete database $\mathcal{I}$ over U is $\mathcal{RA}$*-definable* if there exists a relational algebra query q such that $\mathcal{I} = q(\mathcal{Z}_V)$ for some V.

$\mathcal{RA}$-definability (2)

Theorem

If $\mathcal{I}$ is representable by some c-table T, then $\mathcal{I}$ is $\mathcal{RA}$-definable.

Proof.

Let $\alpha(T) = U = A_1 \dots A_n$. Let $x_1, \dots, x_k$ denote the variables in T and let $V = B_1 \dots B_k$ be a set of attributes such that $\text{dom}(B_j) = \text{dom}(x_j)$. Consider the query

$$q(Z) = \bigcup_{t \in T} \pi_U \left(\sigma_{\rho_{x_1 \dots x_k \to B_1 \dots B_k}(t.con)} \left[A_1(t) \bowtie \cdots \bowtie A_n(t) \bowtie Z \right] \right),$$

where

$$A_i(t) = \begin{cases} \{\, \langle A_i : a \rangle \,\} & \text{if } t.A_i = a \\ \rho_{B_j \to A_i}(\pi_{B_j}(Z)) & \text{if } t.A_i = x_j \end{cases}$$

We have $q(\mathcal{Z}_V) = \mathcal{I}$. □

$\mathcal{RA}$-definability (3)

Example

T

A_1	A_2	*con*
a_1	b_1	$x = 1$
a_2	b_1	$x \neq 1$
a_3	b_2	$y = 1 \wedge x \neq 1$
a_4	b_2	$y \neq 1 \vee x = 1$

$$\mathcal{Z}_V = \left\{ \begin{array}{|c|c|}\hline B_1 & B_2 \\ \hline 1 & 1 \\ \hline\end{array}, \begin{array}{|c|c|}\hline B_1 & B_2 \\ \hline 1 & 2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline B_1 & B_2 \\ \hline 2 & 1 \\ \hline\end{array}, \begin{array}{|c|c|}\hline B_1 & B_2 \\ \hline 2 & 2 \\ \hline\end{array}, \dots \right\}$$

$$\begin{aligned} q(Z) := \quad & \pi_{A_1A_2}\left(\sigma_{B_1=1}\left[\begin{array}{|c|c|}\hline A_1 & A_2 \\ \hline a_1 & b_1 \\ \hline\end{array} \bowtie Z\right]\right) \\ \cup \quad & \pi_{A_1A_2}\left(\sigma_{B_1\neq 1}\left[\begin{array}{|c|c|}\hline A_1 & A_2 \\ \hline a_2 & b_1 \\ \hline\end{array} \bowtie Z\right]\right) \\ \cup \quad & \pi_{A_1A_2}\left(\sigma_{B_2=1\wedge B_1\neq 1}\left[\begin{array}{|c|c|}\hline A_1 & A_2 \\ \hline a_3 & b_2 \\ \hline\end{array} \bowtie Z\right]\right) \\ \cup \quad & \pi_{A_1A_2}\left(\sigma_{B_2\neq 1\vee B_1=1}\left[\begin{array}{|c|c|}\hline A_1 & A_2 \\ \hline a_4 & b_2 \\ \hline\end{array} \bowtie Z\right]\right) \end{aligned}$$

$\mathcal{RA}$-completeness

Definition

A representation system is $\mathcal{RA}$*-complete* if it can represent any $\mathcal{RA}$-definable incomplete database.

Theorem

c-tables are $\mathcal{RA}$*-complete.*

Proof.

Let $\mathcal{I}$ be $\mathcal{RA}$-definable using query $q(\mathcal{Z}_V)$. Let T be a c-table representing $\mathcal{Z}_V$, i.e., set

$T =$

B_1	B_2	...	B_k	*con*
x_1	x_2	...	x_k	true

Since c-tables are closed under $\mathcal{RA}$, $\bar{q}(T)$ produces a c-table that represents $\mathcal{I}$. □

Finite completeness (1)

Definition

A representation system is *finitely complete* if it can represent any finite incomplete database.

Theorem

Boolean c-tables (and hence finite-domain and standard c-tables) are finitely complete.

Corollary

Every $\mathcal{RA}$-complete representation system is finitely complete.

Finite completeness (2)

Proof.

Let $\mathcal{I} = \{ I^0, \ldots, I^{n-1} \}$ be a finite incomplete database and assume wlog that $n = 2^m$ for some positive integer m. Let $\mathbf{x} = (x_{m-1}, \ldots, x_0)$ be a vector of boolean variables. There are 2^m possible values of $\mathbf{x}$; assign a unique one to each I^w, $w \in \{ 0, \ldots, n-1 \}$. Let $c_w(\mathbf{x})$ be a Boolean formula that checks whether $\mathbf{x}$ takes the value assigned to I^w. Then set

$$T[] = \bigcup_w I^w$$

$$T(t) = \bigvee_{w \text{ s.t. } t \in I^w} c_w(\mathbf{x}).$$

We have $\mathrm{Mod}(T) = \mathcal{I}$. □

Finite completeness (3)

Example

$$\mathcal{I} = \left\{ \begin{array}{c} I^0 \\ \begin{array}{|c|c|} \hline A & B \\ \hline a_1 & b_1 \\ & \\ \hline \end{array} \end{array}, \begin{array}{c} I^1 \\ \begin{array}{|c|c|} \hline A & B \\ \hline a_2 & b_2 \\ a_3 & b_3 \\ \hline \end{array} \end{array}, \begin{array}{c} I^2 \\ \begin{array}{|c|c|} \hline A & B \\ \hline a_1 & b_1 \\ a_2 & b_2 \\ \hline \end{array} \end{array}, \begin{array}{c} I^3 \\ \begin{array}{|c|c|} \hline A & B \\ \hline & \\ & \\ \hline \end{array} \end{array} \right\}$$

Instance	$\mathbf{x} = (x_1, x_0)$	$c_w(\mathbf{x})$
I^0	(F, F)	$\neg x_1 \wedge \neg x_0$
I^1	(F, T)	$\neg x_1 \wedge x_0$
I^2	(T, F)	$x_1 \wedge \neg x_0$
I^3	(T, T)	$x_1 \wedge x_0$

$T =$

A	B	con
a_1	b_1	$(\neg x_1 \wedge \neg x_0) \vee (x_1 \wedge \neg x_0)$
a_2	b_2	$(\neg x_1 \wedge x_0) \vee (x_1 \wedge \neg x_0)$
a_3	b_3	$(\neg x_1 \wedge x_0)$

Incompleteness results

Theorem

Codd tables, v-tables, finite-domain Codd tables, finite-domain v-tables, ?-tables, or-set tables, and ?-or-set tables are not finitely complete (and thus not $\mathcal{RA}$-complete).

Proof.

By counterexample. Consider the finite incomplete database

$$\mathcal{I} = \left\{ \begin{array}{|c|c|} \hline A_1 & A_2 \\ \hline a_1 & a_1 \\ \hline \end{array}, \begin{array}{|c|c|} \hline A_1 & A_2 \\ \hline a_2 & a_3 \\ \hline \end{array} \right\}.$$

$\square$

Due to their simplicity (and completion properties), these representation systems are very useful in practice. This motivates the study of weak representation systems.

A note on compactness

In practice, compactness of representation is important!

Example

Let $x_1, \ldots, x_k$ be variables with domain $\{1, 2, \ldots, n\}$. Consider the finite-domain v-table

A_1	A_2	$\ldots$	A_k
x_1	x_2	$\ldots$	x_k

.

The corresponding Boolean c-table has n^k rows!

Outline

Certain answer tuple semantics (1)

Definition

Let $\mathcal{I}$ be an incomplete database and q a relational algebra query. The *q-information* $\mathcal{I}^q$ is given by the set of certain tuples in $q(\mathcal{I})$, i.e., $\mathcal{I}^q = \cap_{I \in q(\mathcal{I})} I$. Note that $\mathcal{I}^q$ is a certain database; it constitutes the query result under the *certain answer tuple semantics*.

Example

- $\mathcal{I} = \left\{ \overset{I^1}{\boxed{\begin{array}{c|c} \text{Anna} & \text{Guan} \\ \text{Bob} & \text{Guan} \end{array}}}, \overset{I^2}{\boxed{\begin{array}{c|c} \text{Anna} & \text{Guan} \\ \text{Bob} & \text{Hb} \end{array}}} \right\}$
- $\mathcal{I}^R = I^1 \cap I^2 = \boxed{\begin{array}{c|c} \text{Anna} & \text{Guan} \end{array}}$
- $\mathcal{I}^{\pi_S(R)} = \pi_S(I^1) \cap \pi_S(I^2) = \boxed{\text{Guan}}$
- $\mathcal{I}^{\pi_N(R)} = \pi_N(I^1) \cap \pi_N(I^2) = \boxed{\begin{array}{c} \text{Anna} \\ \text{Bob} \end{array}}$

Different relational queries expose more or less information about certain tuples!

Certain answer tuple semantics (2)

Definition

Let T be a table and q a relational algebra query. The *q-information* T^q is given by the set of certain tuples in $q(\mathrm{Mod}(\mathcal{I}))$, i.e., $T^q = \bigcap_{I \in q(\mathrm{Mod}\,(\mathcal{I}))} I$. Note that T^q is a certain database.

Example

Suppose $\mathrm{dom}(x) = \{\,\mathsf{A}, \mathsf{B}\,\}$ and $\mathrm{dom}(y) = \{\,\mathsf{G}, \mathsf{H}\,\}$.

$$\mathrm{Mod}\left(\overset{T}{\boxed{\begin{array}{c|c} \mathsf{A} & y \\ x & \mathsf{H} \end{array}}}\right) = \left\{ \boxed{\begin{array}{c|c} \mathsf{A} & \mathsf{G} \\ \mathsf{A} & \mathsf{H} \end{array}}, \boxed{\begin{array}{c|c} \mathsf{A} & \mathsf{G} \\ \mathsf{B} & \mathsf{H} \end{array}}, \boxed{\begin{array}{c|c} \mathsf{A} & \mathsf{H} \end{array}}, \boxed{\begin{array}{c|c} \mathsf{A} & \mathsf{H} \\ \mathsf{B} & \mathsf{H} \end{array}} \right\}$$

- $T^R = \emptyset$
- $T^{\pi_N(R)} = \{\,\mathsf{A}\,\}$
- $T^{\pi_S(R)} = \{\,\mathsf{H}\,\}$

Intuition: Uncertain tuples that remain after “applying” q are omitted.

$\mathscr{L}$-equivalency

Definition

Two sets of incomplete databases $\mathcal{I}$ and $\mathcal{J}$ are $\mathscr{L}$*-equivalent*, denoted $\mathcal{I} \equiv_{\mathscr{L}} \mathcal{J}$ if $\mathcal{I}^q = \mathcal{J}^q$ for all $\mathscr{L}$-expressions q.

Example

$$\mathcal{I} = \left\{ \begin{array}{|c|c|} \hline \text{Anna} & \text{Guan} \\ \text{Bob} & \text{Hum. bird} \\ \hline \end{array}, \begin{array}{|c|c|} \hline \text{Anna} & \text{Guan} \\ \text{Bob} & \text{Kingfisher} \\ \hline \end{array} \right\}$$

$$\mathcal{J} = \left\{ \begin{array}{|c|c|} \hline \text{Anna} & \text{Guan} \\ \hline \end{array} \right\}$$

- $\mathcal{I}$ and $\mathcal{J}$ are $\emptyset$-equivalent
- But: $\mathcal{I}$ and $\mathcal{J}$ are *not* P-equivalent (consider π_A)

$\mathscr{L}$-equivalent databases are indistinguishable w.r.t. the certain tuples in the query result.

More examples of $\mathscr{L}$-equivalency

Example

$$\mathcal{I} = \left\{ \boxed{a_1 \mid b_1 \mid c_1}, \begin{array}{|c|c|c|}\hline a_1 & b_2 & c_2 \\ a_2 & b_1 & c_2 \\ \hline\end{array} \right\}$$

$$\mathcal{J} = \left\{ \boxed{a_1 \mid b_1 \mid c_1}, \begin{array}{|c|c|c|}\hline a_1 & b_2 & c_2 \\ a_2 & b_1 & c_3 \\ \hline\end{array} \right\}$$

- $\mathcal{I}$ and $\mathcal{J}$ are $\emptyset$-equivalent
- $\mathcal{I}$ and $\mathcal{J}$ are P-equivalent
- $\mathcal{I}$ and $\mathcal{J}$ are J-equivalent
- $\mathcal{I}$ and $\mathcal{J}$ are *not* PJ-equivalent; e.g., set

$$q(R) = \pi_{AB}(\pi_{AC}(R) \bowtie \pi_{BC}(R)).$$

Then $a_1 b_1 \in \mathcal{I}^q$ but $a_1 b_1 \notin \mathcal{J}^q$.

Weak representation system

Definition

A representation system is *weak* under a query language $\mathscr{L}$ if for any $\mathscr{L}$-expression q and any table T there is a computable table $\bar{q}(T)$ that $\mathscr{L}$-represents $q(\mathrm{Mod}(T))$.

$$\mathrm{Mod}(\bar{q}(T)) \equiv_{\mathscr{L}} q(\mathrm{Mod}(T)).$$

$$\begin{tikzcd}
T \arrow[rr, "\mathrm{Mod}"] \arrow[dd, "\bar{q}"'] & & \mathrm{Mod}(T) \arrow[d, "q"] \\
& & q(\mathrm{Mod}(T)) \\
\bar{q}(T) \arrow[r, "\mathrm{Mod}"] & \mathrm{Mod}(\bar{q}(T)) \arrow[ur, dash, dashed, "\equiv_{\mathscr{L}}"'] &
\end{tikzcd}$$

Weak representation systems correctly determine the certain tuples under $\mathscr{L}$.

PS on Codd-Tables

Theorem

Codd tables are weak under PS.

$$\bar{\sigma}_P(T) = \{\, t \mid t \in T \textit{ and } P(v(t)) \textit{ for all valuations for } \mathrm{Var}(T) \,\}$$
$$\bar{\pi}_U(T) = \pi_U(T)$$

Example

T

Name	*Species*	*Location*
Anna	Guan	@
@	@	Paris
Bob	Kingf.	@

$\bar{\sigma}_{N=\mathrm{B}}(T)$

N	*S*	*L*
B	K	@

$\bar{\pi}_{NS}(T)$

N	*S*
A	G
@	@
B	K

$\bar{\pi}_{NS}(\bar{\sigma}_{N=\mathrm{B}}(T))$

N	*S*
B	K

These are single-relation queries!

PJ/PSU on Codd-Tables

Theorem

Codd tables are not weak under PJ or PSU.

Proof (for PJ).

- Consider Codd table T and set $\mathcal{I} = \text{Mod}(T)$
- Set $q(R) = \pi_{AC}(R) \bowtie \pi_B(R)$
- c-table $T_{q,c}$ represents $\mathcal{I}_q = q(\text{Mod}(T))$.
- Suppose Codd table T_q PJ-represents $\mathcal{I}_q$
- Consider $q' = \pi_{AC}(\pi_{AB}(R) \bowtie \pi_{BC}(R))$
- For each valuation v, T_q must contain tuples t_1, t_2 s.t. $t_1.A = a_2$, $t_2.C = c_1$, and $v(t_1).B = v(t_2).B$
  1. $t_1 = t_2$, then $a_2 c_1 \in T_q^{\pi_{AC}}$ but $a_2 c_1 \notin \mathcal{I}_q^{\pi_{AC}}$ $\rightarrow \lightning$
  2. $t_1 \neq t_2$, then $t_1.B = t_2.B = b$, then $a_2 b \in T_q^{\pi_{AB}}$ for some b but $\mathcal{I}_q^{\pi_{AB}} = \emptyset$ $\rightarrow \lightning$

T

A	B	C
a_1	x	c_1
a_2	y	c_2

$T_{q,c}$

A	B	C
a_1	x	c_1
a_1	y	c_1
a_2	x	c_2
a_2	y	c_2

$\mathcal{I}_q^{q'}$

A	C
a_1	c_1
a_1	c_2
a_2	c_1
a_2	c_2

$\square$

Null values in SQL

SQL null semantics is related but not equal to Codd tables $\rightarrow$ Be careful!

Example

On PostgreSQL.

- $\sigma_{B=1}(T) \rightarrow$ `SELECT * FROM T WHERE B=1`
- $\pi_{AC}(T) \rightarrow$ `SELECT DISTINCT A, C FROM T`

T

A	B	C
1	null	1
2	null	2

$\sigma_{B=1}(T)$

A	B	C

$\sigma_{B\neq 1}(T)$

A	B	C

$\sigma_{B=1 \vee B\neq 1}(T)$

A	B	C

$\pi_{AC}(T)$

A	C
1	1
2	2

$\pi_{B}(T)$

B
null

$T_q = \pi_{AC}(T) \bowtie \pi_B(T)$

A	B	C
1	null	1
2	null	2

$T_q^{q'} = \pi_{AC}(\pi_{AB}(T_q) \bowtie \pi_{BC}(T_q))$

A	C

Positive $\mathcal{RA}$ on v-Tables

Theorem

v-tables are weak under the positive $\mathcal{RA}$. To obtain $\bar{q}$, simply treat variables as distinct constants and use standard $\mathcal{RA}$ operators.

Example

Sightings

N	S
A	G
A	H
z_1	K
z_2	L

VIPs

N
A
B
z_1

$\bar{\sigma}_{N=\mathrm{A}}(\mathrm{S})$

N	S
A	G
A	H

$\mathrm{S} \,\bar{\bowtie}\, \mathrm{V}$

N	S
A	G
A	H
z_1	K

$\bar{\pi}_S(\mathrm{S} \,\bar{\bowtie}\, \mathrm{V})$

S
G
H
K

Easy to do in an off-the-shelf relational database system!

PS$^-$ on v-tables

Theorem

v-tables are not weak under PS$^-$.

Proof.

- Consider v-table T and set $\mathcal{I} = \text{Mod}(T)$
- Set $q(R) = \sigma_{(A=a_1 \wedge B=b) \vee (A=a_2 \wedge B \neq b)}(R)$
- c-table $T_{q,c}$ represents $\mathcal{I}_q = q(\text{Mod}(T))$.
- Suppose v-table T_q PS$^-$-represents $\mathcal{I}_q$
- Consider $q'(R) = \pi_C(\sigma_{A=a_1 \vee A=a_2}(R))$
  1. $(\exists t \in T_q)\, t_1.A = a_i$, then $a_i \in T_q^{\pi_A} \rightarrow \lightning$
  2. $(\forall t \in T_q)\, t.A \in \text{Var}(T)$, then $T_q^{q'} = \emptyset \rightarrow \lightning$

T

A	B	C
a_1	x	c
a_2	x	c

$\mathcal{I}_q^{q'}$

C
c

$T_{q,c}$

A	B	C	con
a_1	x	c	$x = b$
a_2	x	c	$x \neq b$

□

Outline

Algebraic Completion

Definition

Let $(\mathscr{T}, \mathrm{Mod})$ be a representation system and $\mathscr{L}$ be a query language. The representation system obtained by *closing* $\mathscr{T}$ *under* $\mathscr{L}$ is the set of tables $\{\,(T, q) \mid T \in \mathscr{T}, q \in \mathscr{L}\,\}$ and function $\mathrm{Mod}(T, q) = q(\mathrm{Mod}(T))$.

Example

No Codd table for $\mathcal{I}$, but closure of f.d. Codd tables under JR suffices.

$$\mathcal{I} = \left\{ \begin{array}{|c|c|}\hline A & B \\ \hline a_1 & a_1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline A & B \\ \hline a_2 & a_2 \\ \hline \end{array} \right\}, \quad T = \begin{array}{|c|}\hline A \\ \hline a_1 \| a_2 \\ \hline \end{array}, \quad q(R) = R \bowtie \rho_{A \to B}(R)$$

- Think of q as a *view* over T
- View result need not be represented directly

Algebraic completion extends the power of a representation system with the power of a query language.

$\mathcal{RA}$-completion for Codd tables

Theorem

The closure of Codd tables under SPJRU is $\mathcal{RA}$-complete.

Proof.

- c-tables are $\mathcal{RA}$-complete
- Every c-table T can be $\mathcal{RA}$-defined by an SPJRU-query q on $\mathcal{Z}_V$ (see slide 46)
- $\mathcal{Z}_V$ can be represented as a Codd table T'

$T' =$

B_1	B_2	...	B_k
@	@	...	@

- $\text{Mod}(T', q) = q(\text{Mod}(T')) = q(\mathcal{Z}_V) = \text{Mod}(T)$

$\square$

Relational databases with views can represent *any* $\mathcal{RA}$-definable database!

$\mathcal{RA}$-completion for v-tables

Theorem

The closure of v-tables under S^+P is $\mathcal{RA}$-complete.

Proof.

Let $T = \{ t_1, \ldots, t_m \}$ be a c-table on $A_1 \ldots A_n$ and let $\mathrm{Var}(T) = \{ x_1, \ldots, x_k \}$. Express T in terms of v-table T' and query q:

$T' =$

A_1	$\ldots$	A_n	B_1	$\ldots$	B_k	C
$t_1.A_1$	$\ldots$	$t_1.A_n$	x_1	$\ldots$	x_k	1
$t_2.A_1$	$\ldots$	$t_2.A_n$	x_1	$\ldots$	x_k	2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$t_m.A_1$	$\ldots$	$t_m.A_n$	x_1	$\ldots$	x_k	m

$$q(R) = \pi_{A_1 \ldots A_n}(\sigma_{\bigvee_{i=1}^{m}(\psi_i \wedge C=i)}(R))$$

where ψ_i is obtained from $t_i.con$ by replacing all variables x_j by the corresponding attribute B_j. □

Finite completion results

Theorem

The following closures are finitely complete:

1. *or-set-tables under PJ,*
2. *finite v-tables under PJ or S^+P,*
3. *?-tables under $\mathcal{RA}$.*

Proof.

Try it yourself. Hints: Don't start with a c-table, but an incomplete database $\mathcal{I}$. You need two tables for cases 1 and 2; case 3 is quite tricky. $\square$

Outline

Lessons learned

- Incomplete databases are sets of possible databases
- Representation systems are concise descriptions of incomplete databases
- Queries can be analyzed in terms of
  1. Possible answer sets (strong representation)
  2. Certain answer tuples (weak representation)
  3. Possible answer tuples (finite i-databases only)
- Codd tables add null values; weak under PS
  $\rightarrow$ Be careful with null values in SQL
- v-tables add variables; weak under positive $\mathcal{RA}$
- c-tables add variables and conditions; strong under $\mathcal{RA}$ and $\mathcal{RA}$-complete
- $\mathcal{RA}$-views on Codd tables are $\mathcal{RA}$-complete $\rightarrow$ key property!

Suggested reading

- Charu C. Aggarwal (Ed.)
  Managing and Mining Uncertain Data (Chapter 2)
  Springer, 2009

- Dan Suciu, Dan Olteanu, Christopher Ré, Christoph Koch
  Probabilistic Databases (Chapter 2)
  Morgan & Claypool, 2011

- Serge Abiteboul, Richard Hull, Victor Vianu
  Foundations of Databases: The Logical Level (Chapter 19)
  Addison Wesley, 1994

- Tomasz Imieliński, Witold Lipski, Jr.
  Incomplete Infomation in Relational Databases
  Journal of the ACM, 31(4), Oct. 1984