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

1. Refresher: Relational Algebra
2. Incomplete Databases
3. Strong representation systems
4. Completeness
5. Weak Representation Systems
6. Completion
7. Summary
**Notation**

- Set of *attributes* \( \mathcal{A} \) (countably infinite, totally ordered)
- *Domain* \( \mathcal{D} \) of values for the attributes (countably infinite)
- Elements of \( \mathcal{D} \) are called *constants*
- Per-attribute domain denoted \( \text{dom}(A) \)
- Set of *relation names* \( \mathcal{R} \), each associated with a finite set of attributes \( \alpha(R) \subset \mathcal{A} \) (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**

\[ \mathcal{A} = \{ A, B, C, D, \ldots \} = ABCD \ldots \]
\[ \mathcal{D} = \{ a_1, b_1, c_1, a_2, \ldots \} \]
\[ \text{dom}(A) = \{ a_1, a_2, \ldots \} \]
\[ \mathcal{R} = \{ R, S, \ldots \} \]
\[ \alpha(R) = ABC; \text{ write } R[ABC] \]
The Named Perspective

- Let $U \subset A$ be a schema
- *Tuple* $t$ over $U$ is a function $t : U \rightarrow \mathcal{D}$ (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* $I$ of database schema $R$ maps each relation name in $R \in R$ to a relation instance $I(R)$

**Example**

<table>
<thead>
<tr>
<th>$R$</th>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>$a_1$</td>
<td>$b_2$</td>
<td>$c_1$</td>
</tr>
<tr>
<td>$t_2$</td>
<td>$a_2$</td>
<td>$b_1$</td>
<td>$c_1$</td>
</tr>
</tbody>
</table>

- $t_1$ is a tuple over $ABC$
- $t_1 = \langle A: a_1, B: b_2, C: c_1 \rangle = a_1 b_2 c_1$
- $\alpha(t_1) = ABC$
- $t_1(A) = t_1.A = a_1$
- $t_1[AB] = a_1 b_2$ 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., $t \in \mathcal{D}^n$
- 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

$$
\begin{array}{|c|c|c|}
\hline
R & t_1: a_1 & b_2 & c_1 \\
\hline
R & t_2: a_2 & b_1 & c_1 \\
\hline
\end{array}
$$

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

For now, we will mostly use the named perspective.
Relational algebra (1)

- Relation name $R$
- Single-tuple, single-attribute constant relations (VALUES clause)
  
  $$ \{ \langle A: a \rangle \} $$
  
  for $A \in \mathcal{A}$, $a \in \text{dom}(A)$
- Selection $\sigma$ (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 $a \in \text{dom}(A)$.

Example

<table>
<thead>
<tr>
<th>$R$</th>
<th>{ $\langle A: a \rangle }$</th>
<th>$\sigma_{A=a_1}(R)$</th>
<th>$\sigma_{A=a_3}(R)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>$B$</td>
<td>$C$</td>
<td>$A$</td>
</tr>
<tr>
<td>$t_1$:</td>
<td>$a_1$</td>
<td>$b_2$</td>
<td>$c_1$</td>
</tr>
<tr>
<td>$t_2$:</td>
<td>$a_2$</td>
<td>$b_1$</td>
<td>$c_2$</td>
</tr>
<tr>
<td>$t_3$:</td>
<td>$a_1$</td>
<td>$b_1$</td>
<td>$c_1$</td>
</tr>
</tbody>
</table>
Relational algebra (2)

- **Projection** \(\pi\) (SELECT DISTINCT clause)

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

for \(U \subseteq \alpha(R)\)

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

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

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

**Example**

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>(\pi_{AC}(R))</th>
<th>(R \bowtie S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>A</td>
</tr>
<tr>
<td>(t_1): a_1</td>
<td>b_2</td>
<td>c_1</td>
<td>a_1</td>
</tr>
<tr>
<td>(t_2): a_2</td>
<td>b_1</td>
<td>c_2</td>
<td>a_3</td>
</tr>
<tr>
<td>(t_3): a_1</td>
<td>b_1</td>
<td>c_1</td>
<td>a_3</td>
</tr>
</tbody>
</table>
Relational algebra (3)

- **Renaming** of attributes $\rho$ (AS clause)

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

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

- Short notation: only list attributes being renamed

---

**Example**

<table>
<thead>
<tr>
<th></th>
<th>$R$</th>
<th>$\rho_{AB \rightarrow CD}(R)$</th>
<th>$\rho_{AB \rightarrow BA}(R)$</th>
<th>$R \bowtie \rho_{B \rightarrow C}(R)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>$a_1$</td>
<td>$a_1$</td>
<td>$a_1$</td>
<td>$a_1$</td>
</tr>
<tr>
<td>$t_2$</td>
<td>$a_2$</td>
<td>$b_2$</td>
<td>$b_2$</td>
<td>$b_2$</td>
</tr>
<tr>
<td>$t_3$</td>
<td>$a_1$</td>
<td>$b_2$</td>
<td>$b_1$</td>
<td>$b_1$</td>
</tr>
<tr>
<td>$t_4$</td>
<td>$a_1$</td>
<td>$b_2$</td>
<td>$b_1$</td>
<td>$b_1$</td>
</tr>
</tbody>
</table>
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)

\[
I - J = \{ t \mid t \in I \land t \not\in J \}
\]

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

**Example**

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>R ( \cup ) S</th>
<th>R ( - ) S</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>t₁:</td>
<td>t₄:</td>
<td>a₁</td>
<td>b₂</td>
</tr>
<tr>
<td>t₂:</td>
<td>t₅:</td>
<td>a₂</td>
<td>b₁</td>
</tr>
<tr>
<td>t₃:</td>
<td>t₆:</td>
<td>a₁</td>
<td>b₁</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Definition**

Let $\mathcal{L} \subseteq \text{SPJRUD}$ be an algebra. An $\mathcal{L}$-expression is any well-formed relational algebra expression composed of only relation names, constant relations, and the operations in $\mathcal{L}$. Algebra $\mathcal{L}$ 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 $a \in \text{dom}(A)$
  - $\sigma_{A=B}(R)$ for $A, B \in \alpha(R)$
  - $A = a$ and $A = B$ are called predicates

- Generalized selection operators extend the class of predicates
- Positive conjunction
  \[ \sigma_{P_1 \land P_2}(R) = \sigma_{P_1}(\sigma_{P_2}(R)) \]
- Positive disjunction ($S^+$)
  \[ \sigma_{P_1 \lor P_2}(R) = \sigma_{P_1}(R) \cup \sigma_{P_2}(R) \]
- 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

1. Refresher: Relational Algebra
2. Incomplete Databases
3. Strong representation systems
4. Completeness
5. Weak Representation Systems
6. Completion
7. Summary
### Examples of incomplete information

<table>
<thead>
<tr>
<th>Certain data</th>
<th>Uncertain data</th>
<th>Tuple-level uncertainty</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paul owns a car.</td>
<td>Paul may own a car.</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Object</td>
<td>Name</td>
</tr>
<tr>
<td>Paul</td>
<td>Car</td>
<td>Paul</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bob works for Yahoo.</th>
<th>Bob works for either Yahoo or Microsoft.</th>
<th>Attribute-level uncertainty</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Company</td>
<td>Name</td>
</tr>
<tr>
<td>Bob</td>
<td>Yahoo</td>
<td>Bob</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mary sighted a finch. Paul sighted a finch.</th>
<th>Mary sighted a finch or a sparrow. Paul sighted what Mary sighted.</th>
<th>Correlations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Bird</td>
<td>Name</td>
</tr>
<tr>
<td>Mary</td>
<td>Finch</td>
<td>Mary</td>
</tr>
<tr>
<td>Paul</td>
<td>Finch</td>
<td>Paul</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Paul’s favorite number is 17.</th>
<th>Paul has a favorite number, but it is unknown.</th>
<th>Infinity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Num</td>
<td>Name</td>
</tr>
<tr>
<td>Paul</td>
<td>17</td>
<td>Paul</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Certain data</th>
<th>Uncertain data</th>
<th>Tuple-level uncertainty</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paul owns a car.</td>
<td>Paul may own a car.</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Object</td>
<td>Name</td>
</tr>
<tr>
<td>Paul</td>
<td>Car</td>
<td>Paul</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Company</th>
<th>Attribute-level uncertainty</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>Microsoft</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Company</td>
<td>Name</td>
</tr>
<tr>
<td>Bob</td>
<td>Microsoft</td>
<td>Bob</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Bird</th>
<th>Correlations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
<td>Finch</td>
<td></td>
</tr>
<tr>
<td>Paul</td>
<td>Finch</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Bird</td>
<td>Name</td>
</tr>
<tr>
<td>Mary</td>
<td>Finch</td>
<td>Mary</td>
</tr>
<tr>
<td>Paul</td>
<td>Finch</td>
<td>Paul</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Num</th>
<th>Infinity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paul</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Num</td>
<td>Name</td>
</tr>
<tr>
<td>Paul</td>
<td>1</td>
<td>Paul</td>
</tr>
</tbody>
</table>

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

\[ \mathcal{N}_U = \{ I \mid I \text{ is a (finite) relation instance over schema } U \} \]

**Definition**

- An *incomplete relation* (i-relation) \( \mathcal{I} \) over \( U \) is a set of possible relation instances over \( U \), i.e., \( \mathcal{I} \subseteq \mathcal{N}_U \).

- An *incomplete database* (i-database) of a database schema \( R \) maps each relation name \( R \in R \) 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 \( U \): \( \mathcal{I} = \mathcal{N}_U \)

- Incomplete databases can be *infinite* even though every relation instance is finite; e.g., \( \{ a_1, a_2, a_3, \ldots \} \)

- \( \mathcal{N}_U \) 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”) \( \mathcal{T} \) whose elements we call *tables*, and a function Mod that associates to each table \( T \in \mathcal{T} \) an incomplete database \( \text{Mod}(T) \).

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

<table>
<thead>
<tr>
<th>SUPPLIER</th>
<th>LOCATION</th>
<th>PRODUCT</th>
<th>QUANTITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
<td>London</td>
<td>Nails</td>
<td>@</td>
</tr>
<tr>
<td>Brown</td>
<td>@</td>
<td>Bolts</td>
<td>@</td>
</tr>
<tr>
<td>Jones</td>
<td>@</td>
<td>Nuts</td>
<td>40,000</td>
</tr>
</tbody>
</table>

**Definition**

An *@-tuple* on $U$ is an extended tuple in which each attribute $A \in U$ takes values in $\text{dom}(A) \cup \{ @ \}$. 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 \( \text{dom}(A) = \{ a_1, a_2 \} \) and \( \text{dom}(B) = \{ b_1, b_2 \} \).

\[
\text{Mod} \begin{pmatrix} a_1 & @ \\ @ & b_2 \end{pmatrix} = \{ \begin{pmatrix} a_1 & b_1 \\ a_1 & b_2 \end{pmatrix}, \begin{pmatrix} a_1 & b_1 \\ a_2 & b_2 \end{pmatrix}, \begin{pmatrix} a_1 & b_2 \\ a_2 & b_2 \end{pmatrix} \}
\]

Let \( R^* \in \text{RHS of the example} \):

- 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 \( S \in \text{MOD}(T) \iff (\exists R) R \in \text{Mod}(T) \land S \supseteq R \).

Example

\[
\text{MOD} \left( \begin{array}{c|c} a_1 & \@ \\ \hline \@ & b_2 \end{array} \right) = \left\{ \begin{array}{c|c|c|c|c} a_1 & b_2 & a_1 & b_1 & a_1 & b_1 & a_1 & b_2 & a_1 & b_2 & a_1 & b_2 \\ \hline \hline a_1 & b_2 & a_1 & b_1 & a_1 & b_1 & a_1 & b_2 & a_1 & b_2 & a_1 & b_2 \\ \hline a_2 & b_1 & a_2 & b_2 & a_2 & b_2 & a_2 & b_1 & a_2 & b_2 & a_2 & b_2 \\ \hline a_2 & b_1 & a_2 & b_2 & a_2 & b_2 & a_2 & b_1 & a_2 & b_2 & a_2 & b_2 \end{array} \right\}
\]
Let $R^* \in \text{RHS}$ of the example:

- There is no certain tuple, i.e., $\exists t \forall R^* t \in R^*$
- 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 \textit{marked null values} or variables
- \( V(A) \) = set of \textit{variables} for attribute \( A \) (countably infinite)
- \( V(A) \cap V(B) = \emptyset \) if \( \text{dom}(A) \neq \text{dom}(B) \); otherwise \( V(A) = V(B) \)

\begin{itemize}
  \item \textbf{Example}
  \begin{table}[h]
  \centering
  \begin{tabular}{|c|c|c|}
  \hline
  Course & Teacher & Weekday \\
  \hline
  Databases & x & Monday \\
  Programming & y & Tuesday \\
  Databases & x & Thursday \\
  FORTRAN & Smith & \( z \) \\
  \hline
  \end{tabular}
  \end{table}
\end{itemize}

\textbf{Definition}

A \textit{\( v \)-tuple} on \( U \) is an extended tuple in which each attribute \( A \in U \) takes values in \( \text{dom}(A) \cup V(A) \). A \textit{\( v \)-table} is a finite set of \( v \)-tuples.
Models of v-tables

Example

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

- $\text{Mod} \left( \begin{array}{c} a_1 \times x \\ y b_2 \end{array} \right) = \{ \begin{array}{c} a_1 b_1 \\ a_2 b_2 \end{array}, \begin{array}{c} a_1 b_1 \\ a_2 b_2 \end{array}, \begin{array}{c} a_1 b_2 \\ a_2 b_2 \end{array} \}$

- $\text{Mod} \left( \begin{array}{c} c_1 z \\ z c_2 \end{array} \right) = \{ \begin{array}{c} c_1 c_1 \\ c_1 c_2 \\ c_1 c_2 \end{array}, \begin{array}{c} c_1 c_1 \\ c_1 c_2 \\ c_1 c_2 \end{array} \}$

- $\text{Mod} \left( \begin{array}{c} z_1 z_2 \end{array} \right) = \{ \begin{array}{c} c_1 c_1 \\ c_1 c_2 \\ c_2 c_1 \\ c_2 c_2 \end{array} \}$

- $\text{Var}(T) = \{ x \mid \text{variable } x \text{ occurs in } T \}$

- Valuation $\nu : \text{Var}(T) \rightarrow \mathcal{D}$ assigns (valid) values to each variable

- $\nu(T)$ is the relation obtained by replacing all variables by their values

- $\text{Mod}(T) = \{ \nu(T) \mid \nu \text{ is a valuation for } \text{Var}(T) \}$

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

<table>
<thead>
<tr>
<th>Supplier</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
<td>London</td>
</tr>
<tr>
<td></td>
<td>New York</td>
</tr>
<tr>
<td></td>
<td>Los Angeles</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Supplier</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
<td>Nails</td>
</tr>
<tr>
<td></td>
<td>Bolts</td>
</tr>
<tr>
<td></td>
<td>Nuts</td>
</tr>
</tbody>
</table>

\[\pi_{\text{Location,Product}}(SL \Join SP)\]

<table>
<thead>
<tr>
<th>Location</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>London</td>
<td>Nails</td>
</tr>
<tr>
<td>New York</td>
<td>Bolts</td>
</tr>
<tr>
<td>Los Angeles</td>
<td>Nuts</td>
</tr>
</tbody>
</table>
c-Tables

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

### Example

<table>
<thead>
<tr>
<th>Supplier</th>
<th>Location</th>
<th>Product</th>
<th>con</th>
</tr>
</thead>
<tbody>
<tr>
<td>x Brown</td>
<td>London</td>
<td>Nails</td>
<td>$x = \text{Smith}$</td>
</tr>
<tr>
<td></td>
<td>New York</td>
<td>Nails</td>
<td>$x \neq \text{Smith}$</td>
</tr>
</tbody>
</table>

### Definition

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

Example

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

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>\text{con}</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a_1)</td>
<td>(b_1)</td>
<td>(x = 1)</td>
</tr>
<tr>
<td>(a_2)</td>
<td>(b_1)</td>
<td>(x \neq 1)</td>
</tr>
<tr>
<td>(a_3)</td>
<td>(b_2)</td>
<td>(y = 1 \land x \neq 1)</td>
</tr>
<tr>
<td>(a_4)</td>
<td>(b_2)</td>
<td>(y \neq 1 \lor x = 1)</td>
</tr>
</tbody>
</table>

\[
\text{Mod} = \left\{ \begin{array}{c} x_1 y_1 \quad x_1 y_2 \quad x_2 y_1 \quad x_2 y_2 \\ \begin{array}{c} a_1 b_1 \quad a_1 b_1 \quad a_2 b_1 \quad a_2 b_1 \\ a_4 b_2 \quad a_4 b_2 \quad a_3 b_2 \quad a_4 b_2 \\ a_1 b_1 \quad a_2 b_1 \quad a_2 b_1 \quad a_4 b_2 \end{array} \end{array} \right. \\
\begin{array}{c} x_1 y_1 \quad x_1 y_2 \quad x_2 y_1 \quad x_2 y_2 \\ \begin{array}{c} a_1 b_1 \quad a_1 b_1 \quad a_2 b_1 \quad a_2 b_1 \\ a_4 b_2 \quad a_4 b_2 \quad a_3 b_2 \quad a_4 b_2 \\ a_1 b_1 \quad a_2 b_1 \quad a_2 b_1 \quad a_4 b_2 \end{array} \end{array} \right. \\
\begin{array}{c} x_1 y_1 \quad x_1 y_2 \quad x_2 y_1 \quad x_2 y_2 \\ \begin{array}{c} a_1 b_1 \quad a_1 b_1 \quad a_2 b_1 \quad a_2 b_1 \\ a_4 b_2 \quad a_4 b_2 \quad a_3 b_2 \quad a_4 b_2 \\ a_1 b_1 \quad a_2 b_1 \quad a_2 b_1 \quad a_4 b_2 \end{array} \end{array} \right. \\
\begin{array}{c} x_1 y_1 \quad x_1 y_2 \quad x_2 y_1 \quad x_2 y_2 \\ \begin{array}{c} a_1 b_1 \quad a_1 b_1 \quad a_2 b_1 \quad a_2 b_1 \\ a_4 b_2 \quad a_4 b_2 \quad a_3 b_2 \quad a_4 b_2 \\ a_1 b_1 \quad a_2 b_1 \quad a_2 b_1 \quad a_4 b_2 \end{array} \end{array} \right. \\
\begin{array}{c} x_1 y_1 \quad x_1 y_2 \quad x_2 y_1 \quad x_2 y_2 \\ \begin{array}{c} a_1 b_1 \quad a_1 b_1 \quad a_2 b_1 \quad a_2 b_1 \\ a_4 b_2 \quad a_4 b_2 \quad a_3 b_2 \quad a_4 b_2 \\ a_1 b_1 \quad a_2 b_1 \quad a_2 b_1 \quad a_4 b_2 \end{array} \end{array} \right. \}

Valuation check conditions: \( \nu(T) = \{ \nu(t[U]) \mid \nu(t(\text{con})) = \text{true} \} \)

\( \text{Mod}(T) = \{ \nu(T) \mid \nu \text{ is a valuation for } \text{Var}(T) \} \)

\( \nu \)-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 $x \in \text{Var}(T)$ is associated with a finite domain $\text{dom}(x)$.

- 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.

\[
\text{Mod} \left( \begin{array}{c|c} a_1 & b_1 \\
 a_1 & b_2 \\
 a_1 & b_1 \end{array} \right) = \left\{ \begin{array}{c|c} a_1 & b_1, \\
a_1 & b_1 \\
a_1 & b_2 \end{array} \right\}
\]

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

\[
\text{Mod} \left( \begin{array}{c|c} a_1 & b_2 \\
 a_1 & b_1 \|
 b_2 \\
 a_2 & b_1 \|
 b_2 \end{array} \right) = \left\{ \begin{array}{c|c|c|c} a_1 & b_2, \\
a_1 & b_1, \\
a_1 & b_2, \\
 a_1 & b_2, \\
 a_1 & b_2, \\
 a_1 & b_2 \end{array} \right\}
\]

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

\[
\text{Mod} \left( \begin{array}{c|c} a_1 & b_1 \\
 a_2 & b_1 \|
 b_2 \end{array} \right) = \left\{ \begin{array}{c|c} a_1 & b_1, \\
a_1 & b_1, \\
a_1 & b_1 \end{array} \right\}
\]

Equivalent to finite-domain Codd tables.
Outline

1. Refresher: Relational Algebra
2. Incomplete Databases
3. Strong representation systems
4. Completeness
5. Weak Representation Systems
6. Completion
7. Summary
Possible answer set semantics

Definition

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

Example

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

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

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 $q(\text{Mod}(T))$.

- If $\bar{q}(T)$ can always be computed from $q$ and $T$, the representation system is called *strong* under the query language.

$$
\begin{align*}
T & \xrightarrow{\text{Mod}} I \\
\bar{q} & \downarrow \quad \Downarrow q \\
\bar{q}(T) & \xrightarrow{\text{Mod}} q(I)
\end{align*}
$$

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

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( a_1 )</td>
<td>( b_1 )</td>
<td>( x = 1 )</td>
</tr>
<tr>
<td>( a_1 )</td>
<td>( b_1 )</td>
<td>( x = 2 )</td>
</tr>
<tr>
<td>( a_2 )</td>
<td>( b_2 )</td>
<td>true</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( a_1 )</td>
<td>( b_1 )</td>
<td>( x = 1 \lor x = 2 )</td>
</tr>
<tr>
<td>( a_2 )</td>
<td>( b_2 )</td>
<td>true</td>
</tr>
</tbody>
</table>

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., $(x = 1 \land y = 1) \lor (x \neq 1 \land y = 1)$ by $y = 1$
2. Remove tuples in which condition is unsatisfiable; e.g., $x = 1 \land x = 2$
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 $t'.\text{con} = t_1.\text{con} \lor \cdots \lor t_k.\text{con}$.

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 RA.

Proof.

Given a RA query $q$, construct $\bar{q}$ by replacing in $q$ the operators $\pi$, $\sigma$, $\bowtie$, $\cup$, 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 $\text{Var}(T)$.

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Species</th>
<th>con</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anna</td>
<td>Guan</td>
<td>x = 1</td>
</tr>
<tr>
<td>Anna</td>
<td>Humming bird</td>
<td>x = 2</td>
</tr>
<tr>
<td>Bob</td>
<td>y</td>
<td>x = 3</td>
</tr>
<tr>
<td>z</td>
<td>Guan</td>
<td>x = 4</td>
</tr>
</tbody>
</table>

\[
\bar{\pi}_{\text{Name}}(\text{Sightings})
\]

<table>
<thead>
<tr>
<th>Name</th>
<th>con</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anna</td>
<td>x = 1 \lor x = 2</td>
</tr>
<tr>
<td>Bob</td>
<td>x = 3</td>
</tr>
<tr>
<td>z</td>
<td>x = 4</td>
</tr>
</tbody>
</table>
**c-Selection**

**Definition**

\[
\bar{\sigma}_P(T)[] = T[]
\]

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

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

**Example**

**Sightings**

<table>
<thead>
<tr>
<th>N</th>
<th>S</th>
<th>con</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>G</td>
<td>x = 1</td>
</tr>
<tr>
<td>A</td>
<td>H</td>
<td>x = 2</td>
</tr>
<tr>
<td>B</td>
<td>y</td>
<td>x = 3</td>
</tr>
<tr>
<td>z</td>
<td>G</td>
<td>x = 4</td>
</tr>
</tbody>
</table>

\[\bar{\sigma}_{\text{Species=Gu}}(\text{Sightings})\]

<table>
<thead>
<tr>
<th>N</th>
<th>S</th>
<th>con</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>G</td>
<td>x = 1 \land \text{true}</td>
</tr>
<tr>
<td>A</td>
<td>H</td>
<td>x = 2 \land \text{false}</td>
</tr>
<tr>
<td>B</td>
<td>y</td>
<td>x = 3 \land y = \text{G}</td>
</tr>
<tr>
<td>z</td>
<td>G</td>
<td>x = 4 \land \text{true}</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>N</th>
<th>S</th>
<th>con</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>G</td>
<td>x = 1</td>
</tr>
<tr>
<td>B</td>
<td>y</td>
<td>x = 3 \land y = \text{G}</td>
</tr>
<tr>
<td>z</td>
<td>G</td>
<td>x = 4</td>
</tr>
</tbody>
</table>
c-Union

**Definition**

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

**Example**

<table>
<thead>
<tr>
<th>Sightings</th>
<th>VIPs</th>
<th>Sightings $\bar{\cup}$ VIPs</th>
<th>$\bar{SU}$V (simplified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>$\text{con}$</td>
<td>$N$</td>
<td>$\text{con}$</td>
</tr>
<tr>
<td>A</td>
<td>$x = 1$</td>
<td>B</td>
<td>$y = 1$</td>
</tr>
<tr>
<td>B</td>
<td>$x = 2$</td>
<td>C</td>
<td>$y = 2$</td>
</tr>
<tr>
<td>C</td>
<td>$x = 3$</td>
<td>z</td>
<td>$y = 3$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>z</td>
<td>$y = 3$</td>
</tr>
</tbody>
</table>
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 $T_2' = \rho_{V \rightarrow V'}(T_2)$ and $U_2' = \alpha(T_2')$.

$$(T_1 \bar{\bowtie}_V V \rightarrow V' T_2)[] = T_1[] \bowtie T_2'[]$$

$$(T_1 \bar{\bowtie}_V V \rightarrow V' T_2)(t) = T_1(t[U_1]) \land T_2'(t[U_2']) \bigwedge_{A \in V} t.A = t.A'$$

$$T_1 \bar{\bowtie} T_2 = \pi_{U_1 \cup U_2} (T_1 \bar{\bowtie}_V V \rightarrow V' T_2').$$
**Example**

<table>
<thead>
<tr>
<th>$N$</th>
<th>$S$</th>
<th>$con$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>G</td>
<td>$x = 1$</td>
</tr>
<tr>
<td>A</td>
<td>H</td>
<td>$x = 2$</td>
</tr>
<tr>
<td>$z_1$</td>
<td>K</td>
<td>$x = 3$</td>
</tr>
<tr>
<td>$z_2$</td>
<td>L</td>
<td>$x = 4$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$N$</th>
<th>$S$</th>
<th>$con$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>G</td>
<td>$x = 1 \land y = 1 \land true$</td>
</tr>
<tr>
<td>A</td>
<td>H</td>
<td>$x = 2 \land y = 1 \land true$</td>
</tr>
<tr>
<td>$z_1$</td>
<td>K</td>
<td>$x = 3 \land y = 1 \land z_1 = A$</td>
</tr>
<tr>
<td>$z_2$</td>
<td>L</td>
<td>$x = 4 \land y = 1 \land z_2 = A$</td>
</tr>
<tr>
<td>A</td>
<td>G</td>
<td>$x = 1 \land y = 2 \land false$</td>
</tr>
<tr>
<td>A</td>
<td>H</td>
<td>$x = 2 \land y = 2 \land false$</td>
</tr>
<tr>
<td>$z_1$</td>
<td>K</td>
<td>$x = 3 \land y = 2 \land z_1 = B$</td>
</tr>
<tr>
<td>$z_2$</td>
<td>L</td>
<td>$x = 4 \land y = 2 \land z_2 = B$</td>
</tr>
<tr>
<td>A</td>
<td>$z_1$</td>
<td>$x = 1 \land y = 3 \land z_1 = A$</td>
</tr>
<tr>
<td>A</td>
<td>$z_1$</td>
<td>$x = 2 \land y = 3 \land z_1 = A$</td>
</tr>
<tr>
<td>$z_1$</td>
<td>$z_1$</td>
<td>$x = 3 \land y = 3 \land z_1 = z_1$</td>
</tr>
<tr>
<td>$z_2$</td>
<td>$z_1$</td>
<td>$x = 4 \land y = 3 \land z_2 = z_1$</td>
</tr>
</tbody>
</table>
c-Join (3)

Example (continued)

<table>
<thead>
<tr>
<th>$N$</th>
<th>$S$</th>
<th>$\text{con}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>G</td>
<td>$x_1$</td>
</tr>
<tr>
<td>A</td>
<td>H</td>
<td>$x_2$</td>
</tr>
<tr>
<td>$z_1$</td>
<td>K</td>
<td>$x_3$</td>
</tr>
<tr>
<td>$z_2$</td>
<td>L</td>
<td>$x_4$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$N$</th>
<th>$\text{con}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>$y_1$</td>
</tr>
<tr>
<td>B</td>
<td>$y_2$</td>
</tr>
<tr>
<td>$z_1$</td>
<td>$y_3$</td>
</tr>
</tbody>
</table>

$\text{Sightings} \bowtie \leftarrow_{N \to N'} \text{VIPs (simplified)}$

<table>
<thead>
<tr>
<th>$N$</th>
<th>$S$</th>
<th>$N'$</th>
<th>$\text{con}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>G</td>
<td>A</td>
<td>$x_1y_1$</td>
</tr>
<tr>
<td>A</td>
<td>H</td>
<td>A</td>
<td>$x_2y_1$</td>
</tr>
<tr>
<td>$z_1$</td>
<td>K</td>
<td>A</td>
<td>$x_3y_1 \land z_1 = A$</td>
</tr>
<tr>
<td>$z_2$</td>
<td>L</td>
<td>A</td>
<td>$x_4y_1 \land z_2 = A$</td>
</tr>
<tr>
<td>$z_1$</td>
<td>K</td>
<td>B</td>
<td>$x_3y_2 \land z_1 = B$</td>
</tr>
<tr>
<td>$z_2$</td>
<td>L</td>
<td>B</td>
<td>$x_4y_2 \land z_2 = B$</td>
</tr>
</tbody>
</table>

$\text{Sightings} \bowtie \leftarrow_{N \to N'} \text{VIPs (simplified)}$

<table>
<thead>
<tr>
<th>$N$</th>
<th>$S$</th>
<th>$\text{con}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>G</td>
<td>$x_1y_1 \lor (x_1y_3 \land z_1 = A)$</td>
</tr>
<tr>
<td>A</td>
<td>H</td>
<td>$x_2y_1 \lor (x_2y_3 \land z_1 = A)$</td>
</tr>
<tr>
<td>$z_1$</td>
<td>K</td>
<td>$(x_3y_1 \land z_1 = A) \lor (x_3y_2 \land z_1 = B) \lor x_3y_3$</td>
</tr>
<tr>
<td>$z_2$</td>
<td>L</td>
<td>$(x_4y_1 \land z_2 = A) \lor (x_4y_2 \land z_2 = B) \lor (x_4y_3 \land z_2 = z_1)$</td>
</tr>
</tbody>
</table>
c-Difference

Definition (c-Table difference)

\[
(T_1 \overline{-} \text{VIPs})[] = T_1[] \\
(T_1 \overline{-} \text{VIPs})(t) = T_1(t) \bigwedge_{t' \in \text{VIPs}} \neg(t = t' \land \text{VIPs}(t'))
\]

Example

<table>
<thead>
<tr>
<th>Sightings</th>
<th>VIPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>A con</td>
<td>A con</td>
</tr>
<tr>
<td>A x1</td>
<td>B y1</td>
</tr>
<tr>
<td>B x2</td>
<td>C y2</td>
</tr>
<tr>
<td>C x3</td>
<td>z y3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sightings \overline{-} \text{VIPs} (simplified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A con</td>
</tr>
<tr>
<td>A x1 \land \neg(z = A \land y3)</td>
</tr>
<tr>
<td>B x2 \land \neg(y1) \land \neg(z = B \land y3)</td>
</tr>
<tr>
<td>C x3 \land \neg(y2) \land \neg(z = C \land y3)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sightings \overline{-} \text{VIPs}</th>
</tr>
</thead>
<tbody>
<tr>
<td>A con</td>
</tr>
<tr>
<td>A x1 \land \neg(false \land y1) \land \neg(false \land y2) \land \neg(z = A \land y3)</td>
</tr>
<tr>
<td>B x2 \land \neg(true \land y1) \land \neg(false \land y2) \land \neg(z = B \land y3)</td>
</tr>
<tr>
<td>C x3 \land \neg(false \land y1) \land \neg(true \land y2) \land \neg(z = C \land y3)</td>
</tr>
</tbody>
</table>

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 \(\mathcal{R.A.}\).*

**Proof.**

By counterexample. Consider:

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

\[
\sigma_{A \neq B} \begin{pmatrix}
A & B \\
\hline
x & y
\end{pmatrix}
\]

where \(\text{dom}(x) = \text{dom}(y)\) and \(|\text{dom}(x)| > 1\).

- \(?-\)tables:

We will see: these systems are still very useful!
Outline

1. Refresher: Relational Algebra
2. Incomplete Databases
3. Strong representation systems
4. Completeness
5. Weak Representation Systems
6. Completion
7. Summary
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 $\mathcal{N}_U$ 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 \( \text{dom}(B_1) = \text{dom}(B_2) = \{1, 2, \ldots \} \).

\[
\mathcal{Z}_V = \left\{ \begin{array}{cc}
B_1 & B_2 \\
1 & 1 \\
B_1 & B_2 \\
1 & 2 \\
B_1 & B_2 \\
2 & 1 \\
B_1 & B_2 \\
2 & 2 \\
\ldots
\end{array} \right\}
\]

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 \( V \).
Theorem

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

Proof.

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

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

where

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

We have $q(Z_V) = \mathcal{I}$. 

\qed
### Example

**$T$**

<table>
<thead>
<tr>
<th>$A_1$</th>
<th>$A_2$</th>
<th>con</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>$b_1$</td>
<td>$x = 1$</td>
</tr>
<tr>
<td>$a_2$</td>
<td>$b_1$</td>
<td>$x \neq 1$</td>
</tr>
<tr>
<td>$a_3$</td>
<td>$b_2$</td>
<td>$y = 1 \land x \neq 1$</td>
</tr>
<tr>
<td>$a_4$</td>
<td>$b_2$</td>
<td>$y \neq 1 \lor x = 1$</td>
</tr>
</tbody>
</table>

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

$$q(Z) := \quad \pi_{A_1 A_2} \left( \sigma_{B_1 = 1} \begin{array}{c|c} A_1 & A_2 \\ a_1 & b_1 \end{array} \times Z \right)$$

$$\cup \quad \pi_{A_1 A_2} \left( \sigma_{B_1 \neq 1} \begin{array}{c|c} A_1 & A_2 \\ a_2 & b_1 \end{array} \times Z \right)$$

$$\cup \quad \pi_{A_1 A_2} \left( \sigma_{B_2 = 1 \land B_1 \neq 1} \begin{array}{c|c} A_1 & A_2 \\ a_3 & b_2 \end{array} \times Z \right)$$

$$\cup \quad \pi_{A_1 A_2} \left( \sigma_{B_2 \neq 1 \lor B_1 = 1} \begin{array}{c|c} A_1 & A_2 \\ a_4 & b_2 \end{array} \times Z \right)$$
**RA-completeness**

**Definition**

A representation system is **RA-complete** if it can represent any **RA**-definable incomplete database.

**Theorem**

*C-tables are RA-complete.*

**Proof.**

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

\[
T = \begin{array}{cccc}
B_1 & B_2 & \ldots & B_k \\
x_1 & x_2 & \ldots & x_k \\
\end{array}
\]

Since c-tables are closed under **RA**, \( \bar{q}(T) \) produces a c-table that represents \( I \).
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 \( \text{Mod}(T) = \mathcal{I} \).
Finite completeness (3)

Example

\[ I = \begin{cases} \mathcal{I}^0 & \mathcal{I}^1 & \mathcal{I}^2 & \mathcal{I}^3 \\ A & B & A & B & A & B & A & B \\ a_1 & b_1 & a_2 & b_2 & a_1 & b_1 & a_2 & b_2 \\ a_3 & b_3 & a_2 & b_2 & & & & \end{cases} \]

<table>
<thead>
<tr>
<th>Instance</th>
<th>( x = (x_1, x_0) )</th>
<th>( c_w(x) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathcal{I}^0 )</td>
<td>((F, F))</td>
<td>( \neg x_1 \land \neg x_0 )</td>
</tr>
<tr>
<td>( \mathcal{I}^1 )</td>
<td>((F, T))</td>
<td>( \neg x_1 \land x_0 )</td>
</tr>
<tr>
<td>( \mathcal{I}^2 )</td>
<td>((T, F))</td>
<td>( x_1 \land \neg x_0 )</td>
</tr>
<tr>
<td>( \mathcal{I}^3 )</td>
<td>((T, T))</td>
<td>( x_1 \land x_0 )</td>
</tr>
</tbody>
</table>

\[ T = \begin{array}{cc|c} A & B & \text{con} \\ \hline a_1 & b_1 & (\neg x_1 \land \neg x_0) \lor (x_1 \land \neg x_0) \\ a_2 & b_2 & (\neg x_1 \land x_0) \lor (x_1 \land \neg x_0) \\ a_3 & b_3 & (\neg x_1 \land x_0) \end{array} \]
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 RA-complete).*

**Proof.**

By counterexample. Consider the finite incomplete database

\[ \mathcal{I} = \left\{ \begin{array}{cc}
A_1 & A_2 \\
 a_1 & a_1 \\
 A_1 & A_2 \\
 a_2 & a_3 
\end{array} \right\}. \]

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

$$
\begin{array}{cccc}
A_1 & A_2 & \cdots & A_k \\
\hline
x_1 & x_2 & \cdots & x_k \\
\end{array}
$$

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

$\mathcal{I}^q = \bigcap_{I \in q(\mathcal{I})} I$. Note that $\mathcal{I}^q$ is a certain database; it constitutes the query result under the certain answer tuple semantics.

Example

$\mathcal{I} = \left\{ \begin{array}{c c c}
I^1 & I^2 \\
\text{Anna} & \text{Guan} & \text{Bob} & \text{Guan} \\
\text{Bob} & \text{Guan} & \text{Bob} & \text{Hb}
\end{array} \right\}$

$\mathcal{I}^R = I^1 \cap I^2 = \text{Anna, Guan}$

$\mathcal{I}^{\pi_S(R)} = \pi_S(I^1) \cap \pi_S(I^2) = \text{Guan}$

$\mathcal{I}^{\pi_N(R)} = \pi_N(I^1) \cap \pi_N(I^2) = \text{Anna, Bob}$

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 $q(\text{Mod}(\mathcal{I}))$, i.e., $T^q = \cap_{I \in q(\text{Mod}(\mathcal{I}))} I$. Note that $T^q$ is a certain database.

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

$T$

$\text{Mod} \begin{pmatrix} A & y \\ x & H \end{pmatrix} = \{ A \ G, A \ G, A \ H, A \ H, B \ H, B \ H \}$

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

Intuition: Uncertain tuples that remain after “applying” $q$ are omitted.
$L$-equivalency

Definition

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

Example

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

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

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

$L$-equivalent databases are indistinguishable w.r.t. the certain tuples in the query result.
More examples of $\mathcal{L}$-equivalency

Example

$I = \left\{ \begin{array}{c|c|c} a_1 & b_1 & c_1 \\ \hline a_2 & b_1 & c_2 \end{array} \right\}, \quad J = \left\{ \begin{array}{c|c|c} a_1 & b_1 & c_1 \\ \hline a_2 & b_1 & c_2 \end{array} \right\}, \quad J = \left\{ \begin{array}{c|c|c} a_1 & b_1 & c_1 \\ \hline a_2 & b_1 & c_3 \end{array} \right\}$

- $I$ and $J$ are $\emptyset$-equivalent
- $I$ and $J$ are P-equivalent
- $I$ and $J$ are J-equivalent
- $I$ and $J$ are not PJ-equivalent; e.g., set $q(R) = \pi_{AB}(\pi_{AC}(R) \Join \pi_{BC}(R))$.

Then $a_1 b_1 \in I^q$ but $a_1 b_1 \notin J^q$. 
A representation system is \textit{weak} under a query language $\mathcal{L}$ if for any $\mathcal{L}$-expression $q$ and any table $T$ there is a computable table $\bar{q}(T)$ that $\mathcal{L}$-represents $q(\text{Mod}(T))$.

\[
\text{Mod}(\bar{q}(T)) \equiv_{\mathcal{L}} q(\text{Mod}(T)).
\]
Theorem

Codd tables are weak under PS.

\[ \bar{\sigma}_P(T) = \{ t \mid t \in T \text{ and } P(v(t)) \text{ for all valuations for } \text{Var}(T) \} \]

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

Example

<table>
<thead>
<tr>
<th>Name</th>
<th>Species</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anna</td>
<td>Guan</td>
<td>@</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Paris</td>
</tr>
<tr>
<td>Bob</td>
<td>Kingf.</td>
<td>@</td>
</tr>
</tbody>
</table>

These are single-relation queries!
Theorem

Codd tables are not weak under PJ or PSU.

Proof (for PJ).

- Consider Codd table $T$ and set $I = \text{Mod}(T)$
- Set $q(R) = \pi_{AC}(R) \times \pi_B(R)$
- $c$-table $T_{q,c}$ represents $I_q = q(\text{Mod}(T))$.
- Suppose Codd table $T_q$ PJ-represents $I_q$
- Consider $q' = \pi_{AC}(\pi_{AB}(R) \times \pi_{BC}(R))$
- For each valuation $\nu$, $T_q$ must contain tuples $t_1, t_2$ s.t. $t_1.A = a_2$, $t_2.C = c_1$, and $\nu(t_1).B = \nu(t_2).B$

1. $t_1 = t_2$, then $a_2c_1 \in T_q^{\pi_{AC}}$ but $a_2c_1 \notin I_q^{\pi_{AC}}$  
   $\rightarrow \bot$

2. $t_1 \neq t_2$, then $t_1.B = t_2.B = b$, then $a_2b \in T_q^{\pi_{AB}}$ for some $b$ but $I_q^{\pi_{AB}} = \emptyset$  
   $\rightarrow \bot$
Null values in SQL

SQL null semantics is related but not equal to Codd tables → 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

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>1</td>
<td>null</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>null</td>
<td>2</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>1</td>
<td>null</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>null</td>
<td>2</td>
</tr>
</tbody>
</table>

$\sigma_{B\neq1}(T)$

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>1</td>
<td>null</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>null</td>
<td>2</td>
</tr>
</tbody>
</table>

$\sigma_{B=1 \lor B\neq1}(T)$

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

$\pi_{AC}(T)$

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>1</td>
<td>null</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>null</td>
<td>2</td>
</tr>
</tbody>
</table>

$\pi_B(T)$

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>1</td>
<td>null</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>null</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>1</td>
<td>null</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>null</td>
<td>2</td>
</tr>
</tbody>
</table>

$T_q' = \pi_{AC}(\pi_{AB}(T_q) \bowtie \pi_{BC}(T_q))$
Positive RA on v-Tables

Theorem

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

Example

<table>
<thead>
<tr>
<th>Sightings</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
</tr>
<tr>
<td>A</td>
</tr>
<tr>
<td>A</td>
</tr>
<tr>
<td>z_1</td>
</tr>
<tr>
<td>z_2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VIPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
</tr>
<tr>
<td>A</td>
</tr>
<tr>
<td>B</td>
</tr>
<tr>
<td>z_1</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>S \times V</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
</tr>
<tr>
<td>A</td>
</tr>
<tr>
<td>A</td>
</tr>
<tr>
<td>z_1</td>
</tr>
</tbody>
</table>

\( \bar{\pi}_S(S \times V) \)

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 $I = \text{Mod}(T)$
- Set $q(R) = (\sigma_{(A=a_1 \land B=b)} \lor (A=a_2 \land B\neq b))(R)$
- c-table $T_{q,c}$ represents $I_q = q(\text{Mod}(T))$.
- Suppose v-table $T_q$ PS$^-\text{-represents}$ $I_q$
- Consider $q'(R) = \pi_C(\sigma_{A=a_1 \lor A=a_2}(R))$
  1. $(\exists t \in T_q) t_1.A = a_i, \text{ then } a_i \in T_{q,\pi A}^{\pi A} \rightarrow \frac{\bot}{\top}$
  2. $(\forall t \in T_q) t.A \in \text{Var}(T), \text{ then } T_{q,q'} = \emptyset \rightarrow \frac{\bot}{\top}$
Outline

1. Refresher: Relational Algebra
2. Incomplete Databases
3. Strong representation systems
4. Completeness
5. Weak Representation Systems
6. Completion
7. Summary
Algebraic Completion

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

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

\[
\mathcal{I} = \left\{ \begin{bmatrix} A & B \\ a_1 & a_1 \end{bmatrix}, \begin{bmatrix} A & B \\ a_2 & a_2 \end{bmatrix} \right\}, \quad T = \begin{bmatrix} A \\ a_1 \parallel a_2 \end{bmatrix}, \quad q(R) = R \bowtie \rho_{A \rightarrow B}(R)
\]

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

**Note:** Algebraic completion extends the power of a representation system with the power of a query language.
**RA-completion for Codd tables**

**Theorem**

The closure of Codd tables under SPJRU is RA-complete.

**Proof.**

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

\[
T' = \begin{array}{cccc}
B_1 & B_2 & \ldots & B_k \\
\emptyset & \emptyset & \ldots & \emptyset \\
\end{array}
\]

- \( \text{Mod}(T', q) = q(\text{Mod}(T')) = q(Z_V) = \text{Mod}(T) \)

Relational databases with views can represent any RA-definable database!
**RA-completion for v-tables**

**Theorem**

*The closure of v-tables under $S^+ P$ is RA-complete.*

**Proof.**

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

\[
T' = \begin{array}{cccccc}
A_1 & \ldots & A_n & B_1 & \ldots & B_k & C \\
 t_1.A_1 & \ldots & t_1.A_n & x_1 & \ldots & x_k & 1 \\
t_2.A_1 & \ldots & t_2.A_n & x_1 & \ldots & x_k & 2 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
t_m.A_1 & \ldots & t_m.A_n & x_1 & \ldots & x_k & m \\
\end{array}
\]

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

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

\[ \square \]
## 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 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.
Outline

1. Refresher: Relational Algebra
2. Incomplete Databases
3. Strong representation systems
4. Completeness
5. Weak Representation Systems
6. Completion
7. Summary
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
  → Be careful with null values in SQL
- v-tables add variables; weak under positive RA
- c-tables add variables and conditions; strong under RA and RA-complete
- RA-views on Codd tables are RA-complete → 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*