Uncertain Data Management
Boolean c-tables

Antoine Amarilli[1], Silviu Maniu[2]

December 5th, 2016

[1]Télécom ParisTech

[2]LRI

c-tables

Remember c-tables:

Member ⋈ Booking

id	date	teacher	class	room	
2	2016-11-28	$\texttt{NULL}_1$	UDM	$\texttt{NULL}_2$	
3	2016-12-05	$\texttt{NULL}_1$	UDM	$\texttt{NULL}_2$	
4	2016-12-12	$\texttt{NULL}_1$	UDM	$\texttt{NULL}_2$	*if* $\texttt{NULL}_0$ *is "UDM"*

c-tables

Remember c-tables:

Member ⋈ Booking

id	date	teacher	class	room	
2	2016-11-28	$\texttt{NULL}_1$	UDM	$\texttt{NULL}_2$	
3	2016-12-05	$\texttt{NULL}_1$	UDM	$\texttt{NULL}_2$	
4	2016-12-12	$\texttt{NULL}_1$	UDM	$\texttt{NULL}_2$	*if* $\texttt{NULL}_0$ *is "UDM"*

→ Variant: Only allow NULLs in the conditions

NULLs in conditions

- The possible tuples are exactly the rows
- Each row may either be kept or deleted
  - → Depends on the condition

`NULL`s in conditions

- The possible tuples are exactly the rows
- Each row may either be kept or deleted
  - → Depends on the condition

→ Finite number of possible worlds

NULLs in conditions

- The possible tuples are exactly the rows
- Each row may either be kept or deleted
  - $\rightarrow$ Depends on the condition

$\rightarrow$ Finite number of possible worlds

- $\rightarrow$ at most 2^N if we have N rows

Example

Member ⋈ Booking

id	date	teacher	class	room	
2	2016-11-28	Antoine	UDM	C42	*if* $\mathtt{NULL}_1$ *is "Antoine"*
3	2016-12-05	Antoine	UDM	C42	*if* $\mathtt{NULL}_1$ *is "Antoine"*
4	2016-12-12	Antoine	UDM	C42	*if* $\mathtt{NULL}_1$ *is "Antoine"*
2	2016-11-28	Silviu	UDM	C42	*if* $\mathtt{NULL}_1$ *is "Silviu"*
3	2016-12-05	Silviu	UDM	C42	*if* $\mathtt{NULL}_1$ *is "Silviu"*
4	2016-12-12	Silviu	UDM	C42	*if* $\mathtt{NULL}_1$ *is "Silviu"*

Table of contents

Definitions

Boolean c-tables

Expressiveness

Boolean c-tables

With Boolean c-tables, we impose:

- the possible values of each $\texttt{NULL}_i$ are `True` and `False`

Boolean c-tables

With Boolean c-tables, we impose:

- the possible values of each $\texttt{NULL}_i$ are `True` and `False`

We can simplify notation

→ We write the NULLs as Boolean variables x_i

→ We replace $x_i = \texttt{True}$ by just x_i

→ We replace $x_i = \texttt{False}$ by $\neg x_i$

Boolean c-tables

With Boolean c-tables, we impose:

- the possible values of each $\texttt{NULL}_i$ are `True` and `False`

We can simplify notation

→ We write the NULLs as Boolean variables x_i

→ We replace $x_i = \texttt{True}$ by just x_i

→ We replace $x_i = \texttt{False}$ by $\neg x_i$

→ We can rewrite:

- $x_i = x_j$ to $(x_i \land x_j) \lor (\neg x_i \land \neg x_j)$
- $x_i \neq x_j$ to $(x_i \land \neg x_j) \lor (\neg x_i \land x_j)$

Boolean c-tables

With Boolean c-tables, we impose:

- the possible values of each $\texttt{NULL}_i$ are `True` and `False`

We can simplify notation

→ We write the `NULL`s as Boolean variables x_i

→ We replace $x_i = \texttt{True}$ by just x_i

→ We replace $x_i = \texttt{False}$ by $\neg x_i$

→ We can rewrite:

  - $x_i = x_j$ to $(x_i \land x_j) \lor (\neg x_i \land \neg x_j)$
  - $x_i \neq x_j$ to $(x_i \land \neg x_j) \lor (\neg x_i \land x_j)$

→ The conditions become Boolean expressions

Expressiveness

Theorem

We can always rewrite a c-table having `NULL`s only in conditions to a Boolean c-table.

Expressiveness

Theorem

We can always rewrite a c-table having `NULL`s only in conditions to a Boolean c-table.

Proof: Two steps:

1. We can pick the `NULL`s in a finite domain
2. We can rewrite any finite domain to `True` and `False`

Step 1: Reducing to a finite domain

- We can choose among infinitely many values for the `NULL`s
- However, the values only appear in the conditions:
  - $\texttt{NULL}_i = \texttt{NULL}_j$
  - $\texttt{NULL}_i$ = “c”
  - Boolean combinations

Step 1: Reducing to a finite domain

- We can choose among infinitely many values for the `NULL`s
- However, the values only appear in the conditions:
  - $\texttt{NULL}_i = \texttt{NULL}_j$
  - $\texttt{NULL}_i =$ “c”
  - Boolean combinations
- We call two assignments of values to `NULL`s equivalent if all conditions evaluate to the same

Step 1: Reducing to a finite domain (example)

→ Call two assignments of values to `NULL`s equivalent if all conditions evaluate to the same.

Consider the following:

- $\texttt{NULL}_1 = \texttt{NULL}_2$
- $\texttt{NULL}_2 =$ “c”

→ E.g.: The assignment (a, d) is equivalent to (b, d)

Step 1: Reducing to a finite domain (example)

→ Call two assignments of values to `NULL`s **equivalent** if all conditions evaluate to the **same**.

Consider the following:

- $\texttt{NULL}_1 = \texttt{NULL}_2$
- $\texttt{NULL}_2$ = “c”

→ E.g.: The assignment (a, d) is equivalent to (b, d)

→ What are the possible assignments?

Step 1: Reducing to a finite domain (example)

→ Call two assignments of values to `NULL`s **equivalent** if all conditions evaluate to the **same**.

Consider the following:

- $\texttt{NULL}_1 = \texttt{NULL}_2$
- $\texttt{NULL}_2$ = "c"

→ E.g.: The assignment (a, d) is equivalent to (b, d)

→ What are the possible assignments?

  - (c, c)
    → true, true

Step 1: Reducing to a finite domain (example)

$\rightarrow$ Call two assignments of values to `NULL`s equivalent if all conditions evaluate to the same.

Consider the following:

- $\texttt{NULL}_1 = \texttt{NULL}_2$
- $\texttt{NULL}_2 =$ "c"

$\rightarrow$ E.g.: The assignment (a, d) is equivalent to (b, d)

$\rightarrow$ What are the possible assignments?

- (c, c)

  $\rightarrow$ true, true
- (x, c) with $x \neq c$

  $\rightarrow$ false, true

Step 1: Reducing to a finite domain (example)

→ Call two assignments of values to `NULL`s equivalent if all conditions evaluate to the same.

Consider the following:

- $\texttt{NULL}_1 = \texttt{NULL}_2$
- $\texttt{NULL}_2$ = “c”

→ E.g.: The assignment (a, d) is equivalent to (b, d)

→ What are the possible assignments?

  - (c, c)
    → true, true
  - (x, c) with $x \neq c$
    → false, true
  - (x, x) with $x \neq c$
    → true, false

Step 1: Reducing to a finite domain (example)

→ Call two assignments of values to `NULL`s **equivalent** if all conditions evaluate to the **same**.

Consider the following:

- $\texttt{NULL}_1 = \texttt{NULL}_2$
- $\texttt{NULL}_2 =$ "c"

→ **E.g.:** The assignment (a, d) is **equivalent** to (b, d)

→ What are the possible **assignments?**

  - (c, c)
    → **true**, **true**
  - (x, c) with $x \neq c$
    → false, **true**
  - (x, x) with $x \neq c$
    → **true**, false
  - (y, x) with $x \neq c$ and $y \neq x$
    → false, false

Step 1: Reducing to a finite domain (concluding)

- Consider all constants that appear: $\mathcal{C}$
- Consider N different values $\mathcal{V}$, where N is the number of `NULL`s

Step 1: Reducing to a finite domain (concluding)

- Consider all constants that appear: $\mathcal{C}$
- Consider N different values $\mathcal{V}$, where N is the number of `NULL`s

→ Gives our domain $\mathcal{D} := \mathcal{C} \sqcup \mathcal{V}$

Step 1: Reducing to a finite domain (concluding)

- Consider all constants that appear: $\mathcal{C}$
- Consider N different values $\mathcal{V}$, where N is the number of `NULL`s

→ Gives our domain $\mathcal{D} := \mathcal{C} \sqcup \mathcal{V}$

Lemma

For any c-table with `NULL`s only in conditions,
its set of possible worlds is the same:

- *under the standard semantics*
- *when `NULL`s range over the finite $\mathcal{D}$.*

Step 1: Reducing to a finite domain (concluding)

- Consider all constants that appear: $\mathcal{C}$
- Consider N different values $\mathcal{V}$, where N is the number of `NULL`s

$\rightarrow$ Gives our domain $\mathcal{D} := \mathcal{C} \sqcup \mathcal{V}$

Lemma

For any c-table with `NULL`s only in conditions,
its set of possible worlds is the same:

- *under the standard semantics*
- *when `NULL`s range over the finite $\mathcal{D}$.*

$\rightarrow$ For simplicity, let's pad $\mathcal{D}$ to have exactly 2^k values for some k

Step 2: Rewriting to Boolean variables

- The domain has size 2^k

Step 2: Rewriting to Boolean variables

- The domain has size 2^k
- Write its values in binary
- Encode each $\texttt{NULL}_i$ to variables $x_i^1, \ldots, x_i^k$

Step 2: Rewriting to Boolean variables

- The domain has size 2^k
- Write its values in binary
- Encode each $\texttt{NULL}_i$ to variables $x_i^1, \ldots, x_i^k$

$\rightarrow$ Can we translate the conditions?

Step 2: Rewriting to Boolean variables (example)

- $\texttt{NULL}_7 = 001$

Step 2: Rewriting to Boolean variables (example)

- $\texttt{NULL}_7 = 001$

  $\rightarrow$ $x_7^1 = 0$ and $x_7^2 = 0$ and $x_7^3 = 1$

Step 2: Rewriting to Boolean variables (example)

- $\texttt{NULL}_7 = 001$
  - $\rightarrow$ $x_7^1 = 0$ and $x_7^2 = 0$ and $x_7^3 = 1$
  - $\rightarrow$ $\neg x_7^1 \wedge \neg x_7^2 \wedge x_7^3$

Step 2: Rewriting to Boolean variables (example)

- $\texttt{NULL}_7 = 001$
  - $\rightarrow$ $x_7^1 = 0$ and $x_7^2 = 0$ and $x_7^3 = 1$
  - $\rightarrow$ $\neg x_7^1 \land \neg x_7^2 \land x_7^3$
- $\texttt{NULL}_7 \neq 001$

Step 2: Rewriting to Boolean variables (example)

- $\texttt{NULL}_7 = 001$
  - $\rightarrow$ $x_7^1 = 0$ and $x_7^2 = 0$ and $x_7^3 = 1$
  - $\rightarrow$ $\neg x_7^1 \wedge \neg x_7^2 \wedge x_7^3$
- $\texttt{NULL}_7 \neq 001$
  - $\rightarrow$ negate the above

Step 2: Rewriting to Boolean variables (example)

- $\texttt{NULL}_7 = 001$
  - $\rightarrow$ $x_7^1 = 0$ and $x_7^2 = 0$ and $x_7^3 = 1$
  - $\rightarrow$ $\neg x_7^1 \wedge \neg x_7^2 \wedge x_7^3$
- $\texttt{NULL}_7 \neq 001$
  - $\rightarrow$ negate the above
- $\texttt{NULL}_7 = \texttt{NULL}_8$

Step 2: Rewriting to Boolean variables (example)

- $\texttt{NULL}_7 = 001$
  - $\rightarrow$ $x_7^1 = 0$ and $x_7^2 = 0$ and $x_7^3 = 1$
  - $\rightarrow$ $\neg x_7^1 \wedge \neg x_7^2 \wedge x_7^3$
- $\texttt{NULL}_7 \neq 001$
  - $\rightarrow$ negate the above
- $\texttt{NULL}_7 = \texttt{NULL}_8$
  - $\rightarrow$ $x_7^1 = x_8^1$ and $x_7^2 = x_8^2$ and $x_7^3 = x_8^3$

Step 2: Rewriting to Boolean variables (example)

- $\texttt{NULL}_7 = 001$
  - $\rightarrow$ $x_7^1 = 0$ and $x_7^2 = 0$ and $x_7^3 = 1$
  - $\rightarrow$ $\neg x_7^1 \wedge \neg x_7^2 \wedge x_7^3$
- $\texttt{NULL}_7 \neq 001$
  - $\rightarrow$ negate the above
- $\texttt{NULL}_7 = \texttt{NULL}_8$
  - $\rightarrow$ $x_7^1 = x_8^1$ and $x_7^2 = x_8^2$ and $x_7^3 = x_8^3$
- $\texttt{NULL}_7 \neq \texttt{NULL}_8$

Step 2: Rewriting to Boolean variables (example)

- $\texttt{NULL}_7 = 001$
  - $\rightarrow$ $x_7^1 = 0$ and $x_7^2 = 0$ and $x_7^3 = 1$
  - $\rightarrow$ $\neg x_7^1 \wedge \neg x_7^2 \wedge x_7^3$
- $\texttt{NULL}_7 \neq 001$
  - $\rightarrow$ negate the above
- $\texttt{NULL}_7 = \texttt{NULL}_8$
  - $\rightarrow$ $x_7^1 = x_8^1$ and $x_7^2 = x_8^2$ and $x_7^3 = x_8^3$
- $\texttt{NULL}_7 \neq \texttt{NULL}_8$
  - $\rightarrow$ negate the above

Concluding

1. We have moved to a finite domain
   (without changing the relation)

Concluding

1. We have moved to a finite domain
   (without changing the relation)
2. We have rewritten to Boolean variables
   (we changed the relation)

Concluding

1. We have moved to a finite domain
   (without changing the relation)
2. We have rewritten to Boolean variables
   (we changed the relation)

$\rightarrow$ It suffices to study Boolean c-tables

Table of contents

Definitions

Boolean c-tables

Expressiveness

Strong representation system

- Are Boolean c-tables a strong representation system for relational algebra? …

Strong representation system

- Are Boolean c-tables a strong representation system for relational algebra? …
  $\rightarrow$ Yes!

Strong representation system

- Are Boolean c-tables a strong representation system for relational algebra? ...
  - → Yes!
    - → c-tables are a strong representation system
    - → `NULL`s will never appear by themselves outside of conditions

Capturing all uncertain relations

- Fix a finite set of possible tuples
- A possible world: a subset of the possible tuples
- An finite uncertain relation: (finite) set of possible worlds

Capturing all uncertain relations

- Fix a finite set of possible tuples
- A possible world: a subset of the possible tuples
- An finite uncertain relation: (finite) set of possible worlds

Booking

date	teacher	room
21	Antoine	Saphir
21	Silviu	Saphir
21	Silviu	C47
28	Antoine	Saphir
28	Antoine	C47
28	Silviu	Saphir

Booking

date	teacher	room
21	Antoine	Saphir
21	Silviu	Saphir
21	Silviu	C47
28	Antoine	Saphir
28	Antoine	C47
28	Silviu	Saphir

Capturing all uncertain relations

- Fix a finite set of possible tuples
- A possible world: a subset of the possible tuples
- An finite uncertain relation: (finite) set of possible worlds

Booking

date	teacher	room
21	Antoine	Saphir
21	Silviu	Saphir
21	Silviu	C47
28	Antoine	Saphir
28	Antoine	C47
28	Silviu	Saphir

Booking

date	teacher	room
21	Antoine	Saphir
21	Silviu	Saphir
21	Silviu	C47
28	Antoine	Saphir
28	Antoine	C47
28	Silviu	Saphir

→ Can we capture all finite uncertain relations?

Capturing finite uncertain relations

- Make multiple copies of possible worlds
  so there are 2^k possible worlds
- Write each possible world in binary

Capturing finite uncertain relations

- Make multiple copies of possible worlds so there are 2^k possible worlds
- Write each possible world in binary

00		01		10	
v	**w**	**v**	**w**	**v**	**w**
a	d	a	d	a	d
b	e	b	e	b	e
c	f	c	f	c	f

Capturing finite uncertain relations

- Make multiple copies of possible worlds so there are 2^k possible worlds
- Write each possible world in binary

00

v	**w**
a	d
b	e
c	f

01

v	**w**
a	d
b	e
c	f

10

v	**w**
a	d
b	e
c	f

11

v	**w**
a	d
b	e
c	f

Numbering tuples

For each tuple, write the possible worlds where it appears

Numbering tuples

For each tuple, write the possible worlds where it appears

00

v	w
a	d
b	e
c	f

01

v	w
a	d
b	e
c	f

10

v	w
a	d
b	e
c	f

11

v	w
a	d
b	e
c	f

Numbering tuples

For each tuple, write the possible worlds where it appears

00	
v	**w**
a	d
b	e
c	f

01	
v	**w**
a	d
b	e
c	f

10	
v	**w**
a	d
b	e
c	f

11	
v	**w**
a	d
b	e
c	f

v	**w**				
a	d	00	01	10	11
b	e		01		
c	f		01	10	11

Making a condition

- Create one non-Boolean variable
  - $\rightarrow$ chooses the world
- Obtain a non-Boolean c-table

Making a condition

- Create one non-Boolean variable
  - $\rightarrow$ chooses the world
- Obtain a non-Boolean c-table

v	**w**				
a	d	00	01	10	11
b	e		01		
c	f		01	10	11

Making a condition

- Create one non-Boolean variable
  - $\rightarrow$ chooses the world
- Obtain a non-Boolean c-table

v	**w**				
a	d	00	01	10	11
b	e		01		
c	f		01	10	11

v	**w**	
a	d	$x = 00 \vee x = 01 \vee x = 10 \vee x = 11$
b	e	$x = 01$
c	f	$x = 01 \vee x = 10 \vee x = 11$

Making a Boolean c-table

Use the previous trick to rewrite to a Boolean c-table

Making a Boolean c-table

Use the previous trick to rewrite to a Boolean c-table

v	**w**	
a	d	$x = 00 \vee x = 01 \vee x = 10 \vee x = 11$
b	e	$x = 01$
c	f	$x = 01 \vee x = 10 \vee x = 11$

Making a Boolean c-table

Use the previous trick to rewrite to a Boolean c-table

v	**w**	
a	d	$x = 00 \lor x = 01 \lor x = 10 \lor x = 11$
b	e	$x = 01$
c	f	$x = 01 \lor x = 10 \lor x = 11$

v	**w**	
a	d	$\neg x_1 \land \neg x_2 \lor \neg x_1 \land x_2 \lor x_1 \land \neg x_2 \lor x_1 \land x_2$
b	e	$\neg x_1 \land x_2$
c	f	$\neg x_1 \land x_2 \lor x_1 \land \neg x_2 \lor x_1 \land x_2$

Conclusion

We have studied:

- First:
  - Codd tables with `NULL`s
  - v-tables with named `NULL`s
  - c-tables with named `NULL`s and conditions

Conclusion

We have studied:

- First:
  - Codd tables with `NULL`s
  - v-tables with named `NULL`s
  - c-tables with named `NULL`s and conditions
- Then:
  - c-tables with `NULL`s only in conditions
  - Boolean c-tables: Boolean variables

Conclusion

We have studied:

- First:
  - Codd tables with `NULL`s
  - v-tables with named `NULL`s
  - c-tables with named `NULL`s and conditions
- Then:
  - c-tables with `NULL`s only in conditions
  - Boolean c-tables: Boolean variables

We have shown:

→ Any c-table with `NULL`s only in conditions rewrites to a Boolean c-table

→ Boolean c-tables capture all finite uncertain relations

→ Boolean c-tables are a strong representation system

→ c-tables are a strong representation system

References I

Abiteboul, S., Hull, R., and Vianu, V. (1995).
Foundations of Databases.
Addison-Wesley.
`http://webdam.inria.fr/Alice/pdfs/all.pdf`.

Suciu, D., Olteanu, D., Ré, C., and Koch, C. (2011).
Probabilistic Databases.
Morgan & Claypool.
Unavailable online.