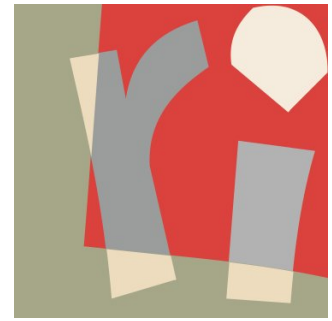

Uncertain Data Management

NULLs

Antoine Amarilli[1], Silviu Maniu[2]

December 5th, 2016

[1]Télécom ParisTech

[2]LRI

Represent missing information in a relation.

Represent missing information in a relation.

Booking

date	teacher	class	room
2016-11-21	Silviu	UDM	Saphir
2016-11-28	Antoine	UDM	Saphir
2016-12-05	Antoine	UDM	NULL
2016-12-12	NULL	UDM	NULL

Represent missing information in a relation.

Booking

date	teacher	class	room
2016-11-21	Silviu	UDM	Saphir
2016-11-28	Antoine	UDM	Saphir
2016-12-05	Antoine	UDM	NULL
2016-12-12	NULL	UDM	NULL

Other name: Codd tables.

Semantics

Each `NULL` can be replaced independently by any domain value

Semantics

Each `NULL` can be replaced independently by any domain value

Booking

date	**teacher**	**class**	**room**
2016-11-21	Silviu	UDM	Saphir
2016-11-28	Antoine	UDM	Saphir
2016-12-05	Antoine	UDM	`NULL`
2016-12-12	`NULL`	UDM	`NULL`

Semantics

Each `NULL` can be replaced independently by any domain value

Booking

date	teacher	class	room
2016-11-21	Silviu	UDM	Saphir
2016-11-28	Antoine	UDM	Saphir
2016-12-05	Antoine	UDM	B543
2016-12-12	Antoine	UDM	Saphir

Semantics

Each `NULL` can be replaced independently by any domain value

Booking

date	teacher	class	room
2016-11-21	Silviu	UDM	Saphir
2016-11-28	Antoine	UDM	Saphir
2016-12-05	Antoine	UDM	xbecz
2016-12-12	gruiiik	UDM	buuuk

Tricky semantics

How can we evaluate queries on Codd tables?

Tricky semantics

How can we evaluate queries on Codd tables?

Booking

date	teacher	class	room
2016-11-21	Silviu	UDM	Saphir
2016-11-28	Antoine	UDM	Saphir
2016-12-05	Antoine	UDM	NULL
2016-12-12	NULL	UDM	NULL

```
SELECT * FROM Booking WHERE teacher='Silviu';
```

Tricky semantics

How can we evaluate queries on Codd tables?

Booking

date	teacher	class	room
2016-11-21	Silviu	UDM	Saphir
2016-11-28	Antoine	UDM	Saphir
2016-12-05	Antoine	UDM	NULL
2016-12-12	NULL	UDM	NULL

```
SELECT * FROM Booking WHERE teacher='Silviu';
```

Booking

date	teacher	class	room
2016-11-21	Silviu	UDM	Saphir

Tricky semantics

How can we evaluate queries on Codd tables?

Booking

date	teacher	class	room
2016-11-21	Silviu	UDM	Saphir
2016-11-28	Antoine	UDM	Saphir
2016-12-05	Antoine	UDM	NULL
2016-12-12	NULL	UDM	NULL

```
SELECT * FROM Booking WHERE teacher='Silviu';
```

Booking

date	teacher	class	room
2016-11-21	Silviu	UDM	Saphir

→ Tricky semantics, often criticized!

Three-valued logic

- Usually, we evaluate operations as Boolean:
  - `WHERE a='42' OR (b=c AND NOT (c=d))`

  → `WHERE False OR (True AND NOT (True))`

  → `False`

Three-valued logic

- Usually, we evaluate operations as Boolean:
  - `WHERE a='42' OR (b=c AND NOT (c=d))`

  → `WHERE False OR (True AND NOT (True))`

  → `False`
- In SQL, values can be True, False, or Unknown (`NULL`)

Three-valued logic

- Usually, we evaluate operations as Boolean:
  - `WHERE a='42' OR (b=c AND NOT (c=d))`

  → `WHERE False OR (True AND NOT (True))`

  → `False`
- In SQL, values can be True, False, or Unknown (`NULL`)
- Essentially anything that involves `NULL` is `NULL`

Three-valued logic (example)

```
WHERE 42=43 OR (42=NULL OR 42=43)
```

Three-valued logic (example)

```
WHERE 42=43 OR (42=NULL OR 42=43)
```

→ `False OR (Unknown OR False)`

Three-valued logic (example)

```
WHERE 42=43 OR (42=NULL OR 42=43)
```

→ `False OR (Unknown OR False)`

→ `False OR Unknown`

Three-valued logic (example)

```
WHERE 42=43 OR (42=NULL OR 42=43)
```

→ `False OR (Unknown OR False)`

→ `False OR Unknown`

→ `Unknown`

Three-valued logic (example)

```
WHERE 42=43 OR (42=NULL OR 42=43)
```

→ False OR (Unknown OR False)

→ False OR Unknown

→ Unknown

```
WHERE 42=45 OR (42=NULL OR 42=42)
```

Three-valued logic (example)

```
WHERE 42=43 OR (42=NULL OR 42=43)
```

→ False OR (Unknown OR False)

→ False OR Unknown

→ Unknown

```
WHERE 42=45 OR (42=NULL OR 42=42)
```

→ False OR (Unknown OR True)

Three-valued logic (example)

```
WHERE 42=43 OR (42=NULL OR 42=43)
```

→ False OR (Unknown OR False)

→ False OR Unknown

→ Unknown

```
WHERE 42=45 OR (42=NULL OR 42=42)
```

→ False OR (Unknown OR True)

→ False OR True

Three-valued logic (example)

```
WHERE 42=43 OR (42=NULL OR 42=43)
```

→ False OR (Unknown OR False)

→ False OR Unknown

→ Unknown

```
WHERE 42=45 OR (42=NULL OR 42=42)
```

→ False OR (Unknown OR True)

→ False OR True

→ True

Three-valued logic (AND table)

AND	True	False
True	True	False
False	False	False

Three-valued logic (AND table)

AND	True	False	NULL
True	True	False	
False	False	False	
NULL			

Three-valued logic (AND table)

AND	True	False	NULL
True	True	False	
False	False	False	False
NULL		False	

Three-valued logic (AND table)

AND	True	False	NULL
True	True	False	NULL
False	False	False	False
NULL	NULL	False	NULL

Three-valued logic (OR table)

OR	True	False
True	True	True
False	True	False

Three-valued logic (OR table)

OR	True	False	NULL
True	True	True	
False	True	False	
NULL			

Three-valued logic (OR table)

OR	True	False	NULL
True	True	True	True
False	True	False	
NULL	True		

Three-valued logic (OR table)

OR	True	False	NULL
True	True	True	True
False	True	False	NULL
NULL	True	NULL	NULL

Three-valued logic (traps)

- What is `NULL * 42`?

Three-valued logic (traps)

- What is `NULL * 42`?

  → `NULL`

Three-valued logic (traps)

- What is `NULL * 42`?

  → `NULL`

- What is `NULL / 0`?

Three-valued logic (traps)

- What is `NULL * 42`?
  - → `NULL`
- What is `NULL / 0`?
  - → Implementation-dependent: `NULL` or error

Three-valued logic (traps)

- What is `NULL * 42`?

  → `NULL`
- What is `NULL / 0`?

  → Implementation-dependent: `NULL` or error
- What is `NULL = NULL`?

Three-valued logic (traps)

- What is `NULL * 42`?

  → `NULL`
- What is `NULL / 0`?

  → Implementation-dependent: `NULL` or error
- What is `NULL = NULL`?

  → `NULL`

Three-valued logic (traps)

- What is `NULL * 42`?

  → `NULL`
- What is `NULL / 0`?

  → Implementation-dependent: `NULL` or error
- What is `NULL = NULL`?

  → `NULL`
- What does the following do?

```
SELECT * FROM Booking WHERE room=NULL
```

Three-valued logic (traps)

- What is `NULL * 42`?
  → `NULL`
- What is `NULL / 0`?
  → Implementation-dependent: `NULL` or error
- What is `NULL = NULL`?
  → `NULL`
- What does the following do?

```
SELECT * FROM Booking WHERE room=NULL
```

  → Returns an empty result

Three-valued logic (traps)

- What is `NULL * 42`?
  → `NULL`
- What is `NULL / 0`?
  → Implementation-dependent: `NULL` or error
- What is `NULL = NULL`?
  → `NULL`
- What does the following do?

```
SELECT * FROM Booking WHERE room=NULL
```

  → Returns an empty result

- What does the following do?

```
SELECT * FROM Booking WHERE room='C42' OR room<>'C42'
```

Three-valued logic (traps)

- What is `NULL * 42`?
  → `NULL`
- What is `NULL / 0`?
  → Implementation-dependent: `NULL` or error
- What is `NULL = NULL`?
  → `NULL`
- What does the following do?

```
SELECT * FROM Booking WHERE room=NULL
```

  → Returns an empty result

- What does the following do?

```
SELECT * FROM Booking WHERE room='C42' OR room<>'C42'
```

  → Return everything where `room` is not `NULL`

Three-valued logic (fixes)

- `IS NULL`

  → test if an expression is `NULL`

- Law of excluded fourth:

  `[COND] IS TRUE OR [COND] IS FALSE OR [COND] IS NULL`

Three-valued logic (more complaints)

This is silly in terms of semantics!

Booking

date	teacher	class	room
2016-12-05	Antoine	UDM	NULL

```
SELECT * FROM Booking WHERE room='C42' OR room<>'C42'
```

Three-valued logic (more complaints)

This is silly in terms of semantics!

Booking

date	teacher	class	room
2016-12-05	Antoine	UDM	NULL

```
SELECT * FROM Booking WHERE room='C42' OR room<>'C42'
```

Possible worlds:

- Either the NULL is 'C42'
- ... or the NULL is something else

Three-valued logic (more complaints)

This is silly in terms of semantics!

Booking

date	teacher	class	room
2016-12-05	Antoine	UDM	NULL

```
SELECT * FROM Booking WHERE room='C42' OR room<>'C42'
```

Possible worlds:

- Either the NULL is 'C42'
- ... or the NULL is something else

→ ... so the tuple should match in either case!

Three-valued logic (more traps!)

Booking

date	teacher	class	room
2016-11-21	Silviu	UDM	Saphir
2016-11-28	Antoine	UDM	Saphir
2016-12-05	Antoine	UDM	`NULL`

Three-valued logic (more traps!)

Booking

date	teacher	class	room
2016-11-21	Silviu	UDM	Saphir
2016-11-28	Antoine	UDM	Saphir
2016-12-05	Antoine	UDM	`NULL`

Repairs

room	cause
C42	lavatory leak
`NULL`	leopard

Three-valued logic (more traps!)

Booking

date	teacher	class	room
2016-11-21	Silviu	UDM	Saphir
2016-11-28	Antoine	UDM	Saphir
2016-12-05	Antoine	UDM	NULL

Repairs

room	cause
C42	lavatory leak
NULL	leopard

```
SELECT * FROM Booking WHERE room NOT IN
  (SELECT room FROM Repairs)
```

Three-valued logic (more traps!)

Booking

date	teacher	class	room
2016-11-21	Silviu	UDM	Saphir
2016-11-28	Antoine	UDM	Saphir
2016-12-05	Antoine	UDM	NULL

Repairs

room	cause
C42	lavatory leak
NULL	leopard

```
SELECT * FROM Booking WHERE room NOT IN
  (SELECT room FROM Repairs)
```

→ Empty result!

Three-valued logic (more traps!)

Booking

date	teacher	class	room
2016-11-21	Silviu	UDM	Saphir
2016-11-28	Antoine	UDM	Saphir
2016-12-05	Antoine	UDM	NULL

Repairs

room	cause
C42	lavatory leak
NULL	leopard

```
SELECT * FROM Booking WHERE room NOT IN
  (SELECT room FROM Repairs)
```

→ Empty result!

```
SELECT * FROM Booking WHERE room NOT IN
  (SELECT room FROM Repairs WHERE room IS NOT NULL)
```

Three-valued logic (more traps!)

Booking

date	teacher	class	room
2016-11-21	Silviu	UDM	Saphir
2016-11-28	Antoine	UDM	Saphir
2016-12-05	Antoine	UDM	NULL

Repairs

room	cause
C42	lavatory leak
NULL	leopard

```
SELECT * FROM Booking WHERE room NOT IN
  (SELECT room FROM Repairs)
```

→ Empty result!

```
SELECT * FROM Booking WHERE room NOT IN
  (SELECT room FROM Repairs WHERE room IS NOT NULL)
```

→ Does not contain the NULL for 2016-12-05

Three-valued logic (more traps!)

Booking

date	teacher	class	room
2016-11-21	Silviu	UDM	Saphir
2016-11-28	Antoine	UDM	Saphir
2016-12-05	Antoine	UDM	NULL

Repairs

room	cause
C42	lavatory leak
NULL	leopard

```
SELECT * FROM Booking WHERE room NOT IN
  (SELECT room FROM Repairs)
```

→ Empty result!

```
SELECT * FROM Booking WHERE room NOT IN
  (SELECT room FROM Repairs WHERE room IS NOT NULL)
```

→ Does not contain the NULL for 2016-12-05

```
SELECT * FROM Booking WHERE
  (room IN (SELECT room FROM Repairs) IS NOT TRUE)
```

Even more traps with NULLs

```
SELECT * FROM R NATURAL JOIN S
```

Even more traps with NULLs

```
SELECT * FROM R NATURAL JOIN S
```

→ NULLs will never join

Even more traps with NULLs

```
SELECT * FROM R NATURAL JOIN S
```

→ NULLs will never join

```
SELECT a FROM R UNION SELECT a FROM S
```

Even more traps with NULLs

```
SELECT * FROM R NATURAL JOIN S
```

→ NULLs will **never** join

```
SELECT a FROM R UNION SELECT a FROM S
```

→ multiple NULLs will **not** be kept

Even more traps with NULLs

```
SELECT * FROM R NATURAL JOIN S
```

→ NULLs will **never** join

```
SELECT a FROM R UNION SELECT a FROM S
```

→ multiple NULLs will **not** be kept

```
SELECT DISTINCT a FROM R
```

Even more traps with NULLs

```
SELECT * FROM R NATURAL JOIN S
```

→ NULLs will never join

```
SELECT a FROM R UNION SELECT a FROM S
```

→ multiple NULLs will not be kept

```
SELECT DISTINCT a FROM R
```

→ multiple NULLs will not be kept

Even, even more traps about NULLs

```
SELECT COUNT(*) FROM R
```

Even, even more traps about NULLs

```
SELECT COUNT(*) FROM R
```

→ NULLs will be counted

Even, even more traps about NULLs

```
SELECT COUNT(*) FROM R
```

→ NULLs will be counted

```
SELECT COUNT(a) FROM R
```

Even, even more traps about NULLs

```
SELECT COUNT(*) FROM R
```

→ NULLs will be counted

```
SELECT COUNT(a) FROM R
```

→ NULLs will be ignored!

Even, even more traps about NULLs

```
SELECT COUNT(*) FROM R
```

→ NULLs will be counted

```
SELECT COUNT(a) FROM R
```

→ NULLs will be ignored!

```
SELECT SUM(a) FROM R
```

Even, even more traps about NULLs

```
SELECT COUNT(*) FROM R
```

→ NULLs will be counted

```
SELECT COUNT(a) FROM R
```

→ NULLs will be ignored!

```
SELECT SUM(a) FROM R
```

→ NULLs will be ignored

Even, even more traps about NULLs

```
SELECT COUNT(*) FROM R
```

→ NULLs will be counted

```
SELECT COUNT(a) FROM R
```

→ NULLs will be ignored!

```
SELECT SUM(a) FROM R
```

→ NULLs will be ignored

```
SELECT AVG(a),SUM(a)/COUNT(*) FROM R
```

Even, even more traps about NULLs

```
SELECT COUNT(*) FROM R
```

→ NULLs will be counted

```
SELECT COUNT(a) FROM R
```

→ NULLs will be ignored!

```
SELECT SUM(a) FROM R
```

→ NULLs will be ignored

```
SELECT AVG(a),SUM(a)/COUNT(*) FROM R
```

→ values may differ

Table of contents

Semantics

- We fix a signature σ:
  - $\rightarrow$ relation names
  - $\rightarrow$ associated arity
- We define uncertain interpretations for each relation

Uncertain relation

- An uncertain relation: set of possible worlds

Uncertain relation

- An uncertain relation: set of possible worlds

Booking			Booking			Booking			
date	**tch**	**room**	**date**	**tch**	**room**	**date**	**tch**	**room**	...
21	S.	a	21	S.	b	21	S.	c	

Relational algebra

- Extend relational algebra operators to uncertain relations
- The possible worlds of the result should be...
  - take all possible worlds of the inputs
  - apply the operation and get a possible output

Relational algebra

- Extend relational algebra operators to uncertain relations
- The possible worlds of the result should be...
  - take all possible worlds of the inputs
  - apply the operation and get a possible output

Booking		
21	S.	a

Booking		
21	S.	b

Booking		
21	S.	c

⋮

Relational algebra

- Extend relational algebra operators to uncertain relations
- The possible worlds of the result should be...
  - take all possible worlds of the inputs
  - apply the operation and get a possible output

Booking		
21	S.	a

Booking		
21	S.	b

$\cup$

Booking		
21	S.	c

$\vdots$

Relational algebra

- Extend relational algebra operators to uncertain relations
- The possible worlds of the result should be...
  - take all possible worlds of the inputs
  - apply the operation and get a possible output

Booking		
21	S.	a

Booking		
21	S.	b

Booking		
21	S.	c

⋮

$\cup$

Booking		
28	a	Saphir

Booking		
28	b	Saphir

Booking		
28	c	Saphir

⋮

Relational algebra

- Extend relational algebra operators to uncertain relations
- The possible worlds of the result should be...
  - take all possible worlds of the inputs
  - apply the operation and get a possible output

Booking
21 S. a

Booking
21 S. b

Booking
21 S. c

⋮

$\cup$

Booking
28 a Saphir

Booking
28 b Saphir

Booking
28 c Saphir

⋮

$=$

Relational algebra

- Extend relational algebra operators to uncertain relations
- The possible worlds of the result should be...
  - take all possible worlds of the inputs
  - apply the operation and get a possible output

Booking		
21	S.	a

Booking		
21	S.	b

Booking		
21	S.	c

⋮

$\cup$

Booking		
28	a	Saphir

Booking		
28	b	Saphir

Booking		
28	c	Saphir

⋮

$=$

Booking		
21	S.	a
28	b	Saphir

Booking		
21	S.	b
28	a	Saphir

⋮

Representation system

Tables with `NULL` are a representation of uncertain tables

Representation system

Tables with `NULL` are a representation of uncertain tables

Booking		
21	S.	`NULL`

Representation system

Tables with `NULL` are a representation of uncertain tables

Booking		
21	S.	`NULL`

stands for

Representation system

Tables with `NULL` are a representation of uncertain tables

Booking		
21	S.	NULL

stands for

Booking		
21	S.	a

Booking		
21	S.	b

Booking		
21	S.	c

⋮

Back to the silly example

Booking

date	teacher	class	room
2016-12-12	Antoine	UDM	NULL
2017-01-09	Silviu	UDM	Sap.

```
SELECT * FROM Booking WHERE teacher='Antoine' AND
  (room='C42' OR room<>'C42')
```

Back to the silly example

Booking

date	teacher	class	room
2016-12-12	Antoine	UDM	NULL
2017-01-09	Silviu	UDM	Sap.

```
SELECT * FROM Booking WHERE teacher='Antoine' AND
  (room='C42' OR room<>'C42')
```

→ How to represent the result?

Representing the output

05	A.	UDM	NULL
12	A.	UDM	Sap.

Representing the output

05	A.	UDM	`NULL`
12	A.	UDM	Sap.

represents

Representing the output

05	A.	UDM	NULL
12	A.	UDM	Sap.

represents

12	A.	UDM	a	12	A.	UDM	b	12	A.	UDM	C42	...
09	S.	UDM	Sap.	09	S.	UDM	Sap.	09	S.	UDM	Sap.	

Representing the output

05	A.	UDM	NULL
12	A.	UDM	Sap.

represents

12	A.	UDM	a	12	A.	UDM	b	12	A.	UDM	C42	...
09	S.	UDM	Sap.	09	S.	UDM	Sap.	09	S.	UDM	Sap.	

```
SELECT * FROM Booking WHERE teacher='A.' AND
  (room='C42' OR room<>'C42')
```

Representing the output

05	A.	UDM	NULL
12	A.	UDM	Sap.

represents

12	A.	UDM	a	12	A.	UDM	b	12	A.	UDM	C42	...
09	S.	UDM	Sap.	09	S.	UDM	Sap.	09	S.	UDM	Sap.	

```
SELECT * FROM Booking WHERE teacher='A.' AND
  (room='C42' OR room<>'C42')
```

12	A.	UDM	a	12	A.	UDM	b	12	A.	UDM	C42	...

Representing the output

05	A.	UDM	NULL
12	A.	UDM	Sap.

represents

12	A.	UDM	a	12	A.	UDM	b	12	A.	UDM	C42	...
09	S.	UDM	Sap.	09	S.	UDM	Sap.	09	S.	UDM	Sap.	

```
SELECT * FROM Booking WHERE teacher='A.' AND
  (room='C42' OR room<>'C42')
```

12	A.	UDM	a	12	A.	UDM	b	12	A.	UDM	C42	...

represented as

Representing the output

05	A.	UDM	NULL
12	A.	UDM	Sap.

represents

12	A.	UDM	a	12	A.	UDM	b	12	A.	UDM	C42	...
09	S.	UDM	Sap.	09	S.	UDM	Sap.	09	S.	UDM	Sap.	

```
SELECT * FROM Booking WHERE teacher='A.' AND
  (room='C42' OR room<>'C42')
```

12	A.	UDM	a	12	A.	UDM	b	12	A.	UDM	C42	...

represented as

12	A.	UDM	NULL

Representation system definition

Uncertain instance: set of possible worlds

Representation system definition

Uncertain instance: set of possible worlds

Uncertainty framework: short way to represent uncertain instances

- Here, Codd tables

Representation system definition

Uncertain instance: set of possible worlds

Uncertainty framework: short way to represent uncertain instances

- Here, Codd tables

Query language: here, relational algebra

Representation system definition

Uncertain instance: set of possible worlds

Uncertainty framework: short way to represent uncertain instances

- Here, Codd tables

Query language: here, relational algebra

Definition (Strong representation system)

For any query in the language,

Representation system definition

Uncertain instance: set of possible worlds

Uncertainty framework: short way to represent uncertain instances

- Here, Codd tables

Query language: here, relational algebra

Definition (Strong representation system)

For any query in the language,
on uncertain instances represented in the framework,

Representation system definition

Uncertain instance: set of possible worlds

Uncertainty framework: short way to represent uncertain instances

- Here, Codd tables

Query language: here, relational algebra

Definition (Strong representation system)
For any query in the language,
on uncertain instances represented in the framework,
the uncertain instance obtained by evaluating the query

Representation system definition

Uncertain instance: set of possible worlds

Uncertainty framework: short way to represent uncertain instances

- Here, Codd tables

Query language: here, relational algebra

Definition (Strong representation system)
For any query in the language,
on uncertain instances represented in the framework,
the uncertain instance obtained by evaluating the query
can also be represented in the framework.

Representation system definition

Uncertain instance: set of possible worlds

Uncertainty framework: short way to represent uncertain instances

- Here, Codd tables

Query language: here, relational algebra

Definition (Strong representation system)
For any query in the language,
on uncertain instances represented in the framework,
the uncertain instance obtained by evaluating the query
can also be represented in the framework.

→ Are Codd tables a strong representation system?

Are Codd tables a representation system?

Member

id	class
1	UDM
2	UDM
3	IE

Booking

date	teacher	class	room
2016-12-05	Antoine	UDM	NULL

Are Codd tables a representation system?

Member

id	class
1	UDM
2	UDM
3	IE

Booking

date	teacher	class	room
2016-12-05	Antoine	UDM	NULL

Can we represent Member $\bowtie$ Booking?

Are Codd tables a representation system?

Member

id	class
1	UDM
2	UDM
3	IE

Booking

date	teacher	class	room
2016-12-05	Antoine	UDM	NULL

Can we represent Member $\bowtie$ Booking?

Member $\bowtie$ Booking

id	date	teacher	class	room
1	2016-12-05	Antoine	UDM	NULL
2	2016-12-05	Antoine	UDM	NULL

Are Codd tables a representation system?

Member

id	class
1	UDM
2	UDM
3	IE

Booking

date	teacher	class	room
2016-12-05	Antoine	UDM	NULL

Can we represent Member $\bowtie$ Booking?

Member $\bowtie$ Booking

id	date	teacher	class	room
1	2016-12-05	Antoine	UDM	NULL
2	2016-12-05	Antoine	UDM	NULL

$\rightarrow$ Can you spot the problem?

Multiple values

- When querying Codd tables, we may duplicate `NULL`s

→ We cannot represent that two `NULL`s are the same

- This may cause problems!

Multiple values example

Booking

date	teacher	class	room
2016-12-12	Antoine	UDM	NULL
2016-01-09	Silviu	UDM	Saphir

$$\Pi_{\textbf{room}}(\text{Booking}) - \Pi_{\textbf{room}}(\sigma_{\textbf{teacher}=\text{“Antoine”}}(\text{Booking}))$$

Multiple values example

Booking

date	teacher	class	room
2016-12-12	Antoine	UDM	`NULL`
2016-01-09	Silviu	UDM	Saphir

$$\Pi_{\textbf{room}}(\text{Booking}) - \Pi_{\textbf{room}}(\sigma_{\textbf{teacher}=\text{“Antoine”}}(\text{Booking}))$$

According to SQL

According to semantics

Saphir

Multiple values example

Booking

date	teacher	class	room
2016-12-12	Antoine	UDM	`NULL`
2016-01-09	Silviu	UDM	Saphir

$$\Pi_{\textbf{room}}(\text{Booking}) - \Pi_{\textbf{room}}(\sigma_{\textbf{teacher}=\text{“Antoine”}}(\text{Booking}))$$

According to SQL	According to semantics
	Saphir

But if we try to represent intermediate expressions?

Multiple values example

Booking

date	teacher	class	room
2016-12-12	Antoine	UDM	NULL
2016-01-09	Silviu	UDM	Saphir

$$\Pi_{\textbf{room}}(\text{Booking}) - \Pi_{\textbf{room}}(\sigma_{\textbf{teacher}=\text{“Antoine”}}(\text{Booking}))$$

According to SQL

According to semantics

Saphir

But if we try to represent intermediate expressions?

$\Pi_{\textbf{room}}(\text{Booking})$

NULL

Saphir

$\Pi_{\textbf{room}}(\sigma_{\textbf{teacher}=\text{“Antoine”}}(\text{Booking}))$

NULL

Table of contents

v-tables

- Idea: give each `NULL` its own name, i.e., named `NULL`s
- Initially, all `NULL`s are distinct
- Propagate their identities

v-tables

- Idea: give each `NULL` its own name, i.e., named `NULL`s
- Initially, all `NULL`s are distinct
- Propagate their identities

Member

id	class
1	UDM
2	UDM
3	IE

Booking

date	teacher	class	room
2016-12-05	$\texttt{NULL}_1$	UDM	$\texttt{NULL}_2$

v-tables

- Idea: give each NULL its own name, i.e., named NULLs
- Initially, all NULLs are distinct
- Propagate their identities

Member

id	class
1	UDM
2	UDM
3	IE

Booking

date	teacher	class	room
2016-12-05	$\texttt{NULL}_1$	UDM	$\texttt{NULL}_2$

Member $\bowtie$ Booking

id	date	teacher	class	room
1	2016-12-05	$\texttt{NULL}_1$	UDM	$\texttt{NULL}_2$
2	2016-12-05	$\texttt{NULL}_1$	UDM	$\texttt{NULL}_2$

v-table semantics

1	2016-12-05	$\texttt{NULL}_1$	UDM	$\texttt{NULL}_2$
2	2016-12-05	$\texttt{NULL}_1$	UDM	$\texttt{NULL}_2$

v-table semantics

1	2016-12-05	$\texttt{NULL}_1$	UDM	$\texttt{NULL}_2$
2	2016-12-05	$\texttt{NULL}_1$	UDM	$\texttt{NULL}_2$

1	2016-12-05	aa	UDM	bb
2	2016-12-05	aa	UDM	bb

v-table semantics

1	2016-12-05	$\texttt{NULL}_1$	UDM	$\texttt{NULL}_2$
2	2016-12-05	$\texttt{NULL}_1$	UDM	$\texttt{NULL}_2$

1	2016-12-05	aa	UDM	bb
2	2016-12-05	aa	UDM	bb

1	2016-12-05	ccc	UDM	ddd
2	2016-12-05	ccc	UDM	ddd

v-table semantics

1	2016-12-05	$\texttt{NULL}_1$	UDM	$\texttt{NULL}_2$
2	2016-12-05	$\texttt{NULL}_1$	UDM	$\texttt{NULL}_2$

1	2016-12-05	aa	UDM	bb
2	2016-12-05	aa	UDM	bb

1	2016-12-05	ccc	UDM	ddd
2	2016-12-05	ccc	UDM	ddd

1	2016-12-05	e	UDM	e
2	2016-12-05	e	UDM	e

Are v-tables tables a representation system?

Member

id	class
1	UDM
2	UDM
3	IE

Booking

date	teacher	class	room
2016-12-05	Antoine	UDM	$\texttt{NULL}_1$

Are v-tables tables a representation system?

Member

id	class
1	UDM
2	UDM
3	IE

Booking

date	teacher	class	room
2016-12-05	Antoine	UDM	$\texttt{NULL}_1$

Can we represent Member $\bowtie$ Booking as a v-table?

Are v-tables tables a representation system?

Member

id	class
1	UDM
2	UDM
3	IE

Booking

date	teacher	class	room
2016-12-05	Antoine	UDM	$\texttt{NULL}_1$

Can we represent Member $\bowtie$ Booking as a v-table?

Member $\bowtie$ Booking

id	date	teacher	class	room
1	2016-12-05	Antoine	UDM	$\texttt{NULL}_1$
2	2016-12-05	Antoine	UDM	$\texttt{NULL}_1$

Are v-tables tables a representation system?

Member

id	class
1	UDM
2	UDM
3	IE

Booking

date	teacher	class	room
2016-12-05	Antoine	UDM	$\texttt{NULL}_1$

Can we represent Member $\bowtie$ Booking as a v-table?

Member $\bowtie$ Booking

id	date	teacher	class	room
1	2016-12-05	Antoine	UDM	$\texttt{NULL}_1$
2	2016-12-05	Antoine	UDM	$\texttt{NULL}_1$

$\rightarrow$ Can you spot the problem?

Are v-tables a representation system? (2)

Member

id	class
1	UDM
2	UDM
3	$\texttt{NULL}_0$

Booking

date	teacher	class	room
2016-12-05	$\texttt{NULL}_1$	UDM	$\texttt{NULL}_2$

Are v-tables a representation system? (2)

Member

id	class
1	UDM
2	UDM
3	$\texttt{NULL}_0$

Booking

date	teacher	class	room
2016-12-05	$\texttt{NULL}_1$	UDM	$\texttt{NULL}_2$

Member $\bowtie$ Booking

id	date	teacher	class	room	
1	2016-12-05	$\texttt{NULL}_1$	UDM	$\texttt{NULL}_2$	
2	2016-12-05	$\texttt{NULL}_1$	UDM	$\texttt{NULL}_2$	
3	2016-12-05	$\texttt{NULL}_1$	UDM	$\texttt{NULL}_2$	*if $\texttt{NULL}_0$ is "UDM"*

Problem

- v-tables cannot represent optional rows
  - $\rightarrow$ the number of rows is certain

Problem

- v-tables cannot represent optional rows
  - → the number of rows is certain
- When selection, join applies to a `NULL`:
  - we do not know how to evaluate
  - we are uncertain about whether the tuple matches

Problem

- v-tables cannot represent optional rows
  - → the number of rows is certain
- When selection, join applies to a `NULL`:
  - we do not know how to evaluate
  - we are uncertain about whether the tuple matches

→ Add conditions to rows!

Condition example

$R := \Pi_{\mathbf{id},\mathbf{room}}(\text{Member} \bowtie \text{Booking})$

id	**room**	*condition*
1	$\texttt{NULL}_2$	
2	$\texttt{NULL}_2$	
3	$\texttt{NULL}_2$	*if* NULL_0 *is "UDM"*

Rooms

room	**seats**
C42	20
$\texttt{NULL}_3$	25

Condition example

$R := \Pi_{\mathbf{id},\mathbf{room}}(\text{Member} \bowtie \text{Booking})$

id	**room**	*condition*
1	$\texttt{NULL}_2$	
2	$\texttt{NULL}_2$	
3	$\texttt{NULL}_2$	*if* $\textit{NULL}_0$ *is "UDM"*

Rooms

room	**seats**
C42	20
$\texttt{NULL}_3$	25

$R \bowtie \text{Rooms}$

id	**room**	**seats**	*condition*

Condition example

$R := \Pi_{\mathbf{id},\mathbf{room}}(\text{Member} \bowtie \text{Booking})$

id	**room**	*condition*
1	$\texttt{NULL}_2$	
2	$\texttt{NULL}_2$	
3	$\texttt{NULL}_2$	*if* NULL_0 *is "UDM"*

Rooms

room	**seats**
C42	20
$\texttt{NULL}_3$	25

$R \bowtie \text{Rooms}$

id	**room**	**seats**	*condition*
1	$\texttt{NULL}_2$	20	*if* NULL_2 *is "C42"*
1	$\texttt{NULL}_2$	25	*if* NULL_2 *is* NULL_3

Condition example

$R := \Pi_{\mathbf{id},\mathbf{room}}(\text{Member} \bowtie \text{Booking})$

id	**room**	*condition*
1	$\texttt{NULL}_2$	
2	$\texttt{NULL}_2$	
3	$\texttt{NULL}_2$	*if* NULL_0 *is "UDM"*

Rooms

room	**seats**
C42	20
$\texttt{NULL}_3$	25

$R \bowtie \text{Rooms}$

id	**room**	**seats**	*condition*
1	$\texttt{NULL}_2$	20	*if* NULL_2 *is "C42"*
1	$\texttt{NULL}_2$	25	*if* NULL_2 *is* NULL_3
2	$\texttt{NULL}_2$	20	*if* NULL_2 *is "C42"*
2	$\texttt{NULL}_2$	25	*if* NULL_2 *is* NULL_3

Condition example

$R := \Pi_{\mathbf{id},\mathbf{room}}(\text{Member} \bowtie \text{Booking})$

id	**room**	*condition*
1	$\texttt{NULL}_2$	
2	$\texttt{NULL}_2$	
3	$\texttt{NULL}_2$	*if* NULL_0 *is "UDM"*

Rooms

room	**seats**
C42	20
$\texttt{NULL}_3$	25

$R \bowtie \text{Rooms}$

id	**room**	**seats**	*condition*
1	$\texttt{NULL}_2$	20	*if* NULL_2 *is "C42"*
1	$\texttt{NULL}_2$	25	*if* NULL_2 *is* NULL_3
2	$\texttt{NULL}_2$	20	*if* NULL_2 *is "C42"*
2	$\texttt{NULL}_2$	25	*if* NULL_2 *is* NULL_3
3	$\texttt{NULL}_2$	20	*if* NULL_2 *is "C42" and* NULL_0 *is "UDM"*
3	$\texttt{NULL}_2$	25	*if* NULL_2 *is* NULL_3 *and* NULL_0 *is "UDM"*

Table of contents

c-tables

- Named `NULL`s, plus conditions on tuples
- Conditions can use:

c-tables

- Named `NULL`s, plus conditions on tuples
- Conditions can use:
  - true
  - false

c-tables

- Named NULLs, plus conditions on tuples
- Conditions can use:
  - true
  - false
  - $\texttt{NULL}_i = \texttt{NULL}_j$

c-tables

- Named NULLs, plus conditions on tuples
- Conditions can use:
  - true
  - false
  - $\texttt{NULL}_i = \texttt{NULL}_j$
  - $\texttt{NULL}_i$ = “value”

c-tables

- Named NULLs, plus conditions on tuples
- Conditions can use:
  - true
  - false
  - $\texttt{NULL}_i = \texttt{NULL}_j$
  - $\texttt{NULL}_i$ = “value”
  - Boolean operators

c-tables

- Named NULLs, plus conditions on tuples
- Conditions can use:
  - true
  - false
  - $\texttt{NULL}_i = \texttt{NULL}_j$
  - $\texttt{NULL}_i$ = “value”
  - Boolean operators

$\rightarrow$ Are c-tables a strong representation system?

Relational algebra operators: product

S	
s	*condition*
s_1	C_1
s_2	C_2

Relational algebra operators: product

S	
s	*condition*
s_1	C_1
s_2	C_2

T	
t	*condition*
t_1	D_1
t_2	D_2

Relational algebra operators: product

S

s	*condition*
s_1	C_1
s_2	C_2

T

t	*condition*
t_1	D_1
t_2	D_2

S × T

s	**t**	*condition*

Relational algebra operators: product

S

s	*condition*
s_1	C_1
s_2	C_2

T

t	*condition*
t_1	D_1
t_2	D_2

S × T

s	**t**	*condition*
s_1	t_1	C_1 *and* D_1
s_1	t_2	C_1 *and* D_2
s_2	t_1	C_2 *and* D_1
s_2	t_2	C_2 *and* D_2

Relational algebra operators: union

S	
s	*condition*
s_0	C_0
s_1	C_1

Relational algebra operators: union

S	
s	*condition*
s_0	C_0
s_1	C_1

S2	
s	*condition*
s_0	D_0
s_2	D_2

Relational algebra operators: union

S	
s	*condition*
s_0	C_0
s_1	C_1

S2	
s	*condition*
s_0	D_0
s_2	D_2

S ∪ S2	
s	*condition*

Relational algebra operators: union

S	
s	*condition*
s_0	C_0
s_1	C_1

S2	
s	*condition*
s_0	D_0
s_2	D_2

S ∪ S2	
s	*condition*
s_0	C_0 *or* D_0
s_1	C_1
s_2	D_2

Relational algebra operators: project

S

s	**t**	*condition*
s_0	t_0	C_0
s_0	t_1	C_1
s_2	t_2	C_2

Relational algebra operators: project

S

s	**t**	*condition*
s_0	t_0	C_0
s_0	t_1	C_1
s_2	t_2	C_2

$\Pi_{\mathbf{s}}(S)$

s	*condition*

Relational algebra operators: project

S

s	**t**	*condition*
s_0	t_0	C_0
s_0	t_1	C_1
s_2	t_2	C_2

$\Pi_{\mathbf{s}}(S)$

s	*condition*
s_0	C_0 *or* C_1
s_2	C_2

Relational algebra operators: select (1)

S

s	**t**	*condition*
42	t_0	C_0
43	t_1	C_1
$\texttt{NULL}_i$	t_2	C_2

Relational algebra operators: select (1)

S

s	**t**	*condition*
42	t_0	C_0
43	t_1	C_1
$\texttt{NULL}_i$	t_2	C_2

$\sigma_{\mathbf{s}=\text{“42”}}(\mathsf{S})$

s	**t**	*condition*

Relational algebra operators: select (1)

S

s	**t**	*condition*
42	t_0	C_0
43	t_1	C_1
$\texttt{NULL}_i$	t_2	C_2

$\sigma_{\mathbf{s}=\text{“42”}}(\mathsf{S})$

s	**t**	*condition*
42	t_0	C_0

Relational algebra operators: select (1)

S

s	**t**	*condition*
42	t_0	C_0
43	t_1	C_1
$\texttt{NULL}_i$	t_2	C_2

$\sigma_{\mathbf{s}=\text{“42”}}(\mathsf{S})$

s	**t**	*condition*
42	t_0	C_0
$\texttt{NULL}_i$	t_2	

Relational algebra operators: select (1)

S

s	**t**	*condition*
42	t_0	C_0
43	t_1	C_1
$\texttt{NULL}_i$	t_2	C_2

$\sigma_{\mathbf{s}=\text{“42”}}(\mathsf{S})$

s	**t**	*condition*
42	t_0	C_0
$\texttt{NULL}_i$	t_2	C_2 *and* NULL_i = “42”

Relational algebra operators: select (2)

S

s	**t**	*condition*
42	42	C_0
43	42	C_1
$\texttt{NULL}_i$	42	C_2
42	$\texttt{NULL}_j$	C_3
$\texttt{NULL}_p$	$\texttt{NULL}_q$	C_4

Relational algebra operators: select (2)

S

s	**t**	*condition*
42	42	C_0
43	42	C_1
$\texttt{NULL}_i$	42	C_2
42	$\texttt{NULL}_j$	C_3
$\texttt{NULL}_p$	$\texttt{NULL}_q$	C_4

$\sigma_{\mathbf{s}=\mathbf{t}}(\mathsf{S})$

s	**t**	*condition*

Relational algebra operators: select (2)

S

s	**t**	*condition*
42	42	C_0
43	42	C_1
$\texttt{NULL}_i$	42	C_2
42	$\texttt{NULL}_j$	C_3
$\texttt{NULL}_p$	$\texttt{NULL}_q$	C_4

$\sigma_{\mathbf{s}=\mathbf{t}}(\mathsf{S})$

s	**t**	*condition*
42	42	

Relational algebra operators: select (2)

S

s	**t**	*condition*
42	42	C_0
43	42	C_1
$\texttt{NULL}_i$	42	C_2
42	$\texttt{NULL}_j$	C_3
$\texttt{NULL}_p$	$\texttt{NULL}_q$	C_4

$\sigma_{\mathbf{s}=\mathbf{t}}(\mathsf{S})$

s	**t**	*condition*
42	42	C_0

Relational algebra operators: select (2)

S

s	**t**	*condition*
42	42	C_0
43	42	C_1
$\texttt{NULL}_i$	42	C_2
42	$\texttt{NULL}_j$	C_3
$\texttt{NULL}_p$	$\texttt{NULL}_q$	C_4

$\sigma_{\mathbf{s}=\mathbf{t}}(S)$

s	**t**	*condition*
42	42	C_0
$\texttt{NULL}_i$	42	

Relational algebra operators: select (2)

S

s	**t**	*condition*
42	42	C_0
43	42	C_1
$\texttt{NULL}_i$	42	C_2
42	$\texttt{NULL}_j$	C_3
$\texttt{NULL}_p$	$\texttt{NULL}_q$	C_4

$\sigma_{\mathbf{s}=\mathbf{t}}(S)$

s	**t**	*condition*
42	42	C_0
$\texttt{NULL}_i$	42	C_2 *and* $\texttt{NULL}_i$ = "42"

Relational algebra operators: select (2)

S

s	**t**	*condition*
42	42	C_0
43	42	C_1
$\texttt{NULL}_i$	42	C_2
42	$\texttt{NULL}_j$	C_3
$\texttt{NULL}_p$	$\texttt{NULL}_q$	C_4

$\sigma_{\mathbf{s}=\mathbf{t}}(S)$

s	**t**	*condition*
42	42	C_0
$\texttt{NULL}_i$	42	C_2 *and* $\texttt{NULL}_i$ = "42"
42	$\texttt{NULL}_j$	

Relational algebra operators: select (2)

S

s	**t**	*condition*
42	42	C_0
43	42	C_1
$\texttt{NULL}_i$	42	C_2
42	$\texttt{NULL}_j$	C_3
$\texttt{NULL}_p$	$\texttt{NULL}_q$	C_4

$\sigma_{\mathbf{s}=\mathbf{t}}(S)$

s	**t**	*condition*
42	42	C_0
$\texttt{NULL}_i$	42	C_2 *and* NULL_i = "42"
42	$\texttt{NULL}_j$	C_3 *and* "42" = NULL_j

Relational algebra operators: select (2)

S

s	**t**	*condition*
42	42	C_0
43	42	C_1
$\texttt{NULL}_i$	42	C_2
42	$\texttt{NULL}_j$	C_3
$\texttt{NULL}_p$	$\texttt{NULL}_q$	C_4

$\sigma_{\mathbf{s}=\mathbf{t}}(S)$

s	**t**	*condition*
42	42	C_0
$\texttt{NULL}_i$	42	C_2 *and* $\texttt{NULL}_i$ = "42"
42	$\texttt{NULL}_j$	C_3 *and* "42" = $\texttt{NULL}_j$
$\texttt{NULL}_p$	$\texttt{NULL}_q$	

Relational algebra operators: select (2)

S

s	**t**	*condition*
42	42	C_0
43	42	C_1
$\texttt{NULL}_i$	42	C_2
42	$\texttt{NULL}_j$	C_3
$\texttt{NULL}_p$	$\texttt{NULL}_q$	C_4

$\sigma_{\mathbf{s}=\mathbf{t}}(S)$

s	**t**	*condition*
42	42	C_0
$\texttt{NULL}_i$	42	C_2 *and* NULL_i = "42"
42	$\texttt{NULL}_j$	C_3 *and* "42" = NULL_j
$\texttt{NULL}_p$	$\texttt{NULL}_q$	C_4 *and* NULL_p = NULL_q

Shortcomings of c-tables

Are c-tables the ultimate uncertainty framework?

- Annotations can become large
  - It may be possible to simplify

Shortcomings of c-tables

Are c-tables the ultimate uncertainty framework?

- Annotations can become large
  - It may be possible to simplify

  $\rightarrow$ In general, this is complicated

Shortcomings of c-tables

Are c-tables the ultimate uncertainty framework?

- Annotations can become large
  - It may be possible to simplify

  $\rightarrow$ In general, this is complicated
- It is intractable to reason about the result!

Shortcomings of c-tables

Are c-tables the ultimate uncertainty framework?

- Annotations can become large
  - It may be possible to simplify

  → In general, this is complicated
- It is intractable to reason about the result!

42	$(\mathtt{NULL}_i = \text{“42”} \text{ and } \mathtt{NULL}_j = \text{“42”}) \text{ or } ((\mathtt{NULL}_k = \mathtt{NULL}_j \text{ or } \mathtt{NULL}_j = \text{“43”}) \text{ and } (\mathtt{NULL}_i = \mathtt{NULL}_j))$

Problems on c-tables

We can represent the output of a query as a c-table

Member $\bowtie$ Booking

id	date	teacher	class	room	
1	2016-12-05	$\texttt{NULL}_1$	UDM	$\texttt{NULL}_2$	
2	2016-12-05	$\texttt{NULL}_1$	UDM	$\texttt{NULL}_2$	
3	2016-12-05	$\texttt{NULL}_1$	UDM	$\texttt{NULL}_2$	*if* $\texttt{NULL}_0$ *is "UDM"*

Problems on c-tables

We can represent the output of a query as a c-table

Member $\bowtie$ Booking

id	date	teacher	class	room	
1	2016-12-05	$\texttt{NULL}_1$	UDM	$\texttt{NULL}_2$	
2	2016-12-05	$\texttt{NULL}_1$	UDM	$\texttt{NULL}_2$	
3	2016-12-05	$\texttt{NULL}_1$	UDM	$\texttt{NULL}_2$	*if* $\texttt{NULL}_0$ *is "UDM"*

What can we ask about it?

Problems on c-tables

We can represent the output of a query as a c-table

Member $\bowtie$ Booking

id	date	teacher	class	room	
1	2016-12-05	$\texttt{NULL}_1$	UDM	$\texttt{NULL}_2$	
2	2016-12-05	$\texttt{NULL}_1$	UDM	$\texttt{NULL}_2$	
3	2016-12-05	$\texttt{NULL}_1$	UDM	$\texttt{NULL}_2$	*if* $\texttt{NULL}_0$ *is "UDM"*

What can we ask about it?

- At the instance level
  - Is an input instance a possible world?
  - Is an input instance the only possible world?

Problems on c-tables

We can represent the output of a query as a c-table

Member $\bowtie$ Booking

id	date	teacher	class	room	
1	2016-12-05	$\texttt{NULL}_1$	UDM	$\texttt{NULL}_2$	
2	2016-12-05	$\texttt{NULL}_1$	UDM	$\texttt{NULL}_2$	
3	2016-12-05	$\texttt{NULL}_1$	UDM	$\texttt{NULL}_2$	*if* $\texttt{NULL}_0$ *is "UDM"*

What can we ask about it?

- At the instance level
  - Is an input instance a possible world?
  - Is an input instance the only possible world?
- At the tuple level
  - Is it possible for an input tuple to be an answer?
  - Is it certain that an input tuple is an answer?

Problems on c-tables

We can represent the output of a query as a c-table

Member $\bowtie$ Booking

id	date	teacher	class	room	
1	2016-12-05	$\texttt{NULL}_1$	UDM	$\texttt{NULL}_2$	
2	2016-12-05	$\texttt{NULL}_1$	UDM	$\texttt{NULL}_2$	
3	2016-12-05	$\texttt{NULL}_1$	UDM	$\texttt{NULL}_2$	*if* NULL_0 *is "UDM"*

What can we ask about it?

- At the instance level
  - Is an input instance a possible world?
  - Is an input instance the only possible world?
- At the tuple level
  - Is it possible for an input tuple to be an answer?
  - Is it certain that an input tuple is an answer?

$\rightarrow$ All intractable in general

Credits

Thanks to Pierre Senellart for useful feedback.

References I


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


Green, T. J. and Tannen, V. (2006).
Models for incomplete and probabilistic information.
IEEE Data Eng. Bull.
http://sites.computer.org/debull/A06mar/green.ps.


Imieliński, T. and Lipski, Jr., W. (1984).
Incomplete information in relational databases.
J. ACM, 31(4).
http://doi.acm.org/10.1145/1634.1886.