



Uncertain Data Management

NULLS

Antoine Amarilli¹, Silviu Maniu²

November 21st, 2017

¹Télécom ParisTech

²LRI

Represent **missing information** in a relation

Represent **missing information** in a relation

Booking

date	teacher	class	room
2017-11-21	Silviu	UDM	Saphir
2017-11-28	Antoine	UDM	Saphir
2017-12-05	Antoine	UDM	NULL
2017-12-12	NULL	UDM	NULL

Represent **missing information** in a relation

Booking

date	teacher	class	room
2017-11-21	Silviu	UDM	Saphir
2017-11-28	Antoine	UDM	Saphir
2017-12-05	Antoine	UDM	NULL
2017-12-12	NULL	UDM	NULL

Other name: Codd tables

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

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

Booking

date	teacher	class	room
2017-11-21	Silviu	UDM	Saphir
2017-11-28	Antoine	UDM	Saphir
2017-12-05	Antoine	UDM	NULL
2017-12-12	NULL	UDM	NULL

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

Booking

date	teacher	class	room
2017-11-21	Silviu	UDM	Saphir
2017-11-28	Antoine	UDM	Saphir
2017-12-05	Antoine	UDM	B543
2017-12-12	Antoine	UDM	Saphir

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

Booking

date	teacher	class	room
2017-11-21	Silviu	UDM	Saphir
2017-11-28	Antoine	UDM	Saphir
2017-12-05	Antoine	UDM	xbecz
2017-12-12	gruiik	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
2017-11-21	Silviu	UDM	Saphir
2017-11-28	Antoine	UDM	Saphir
2017-12-05	Antoine	UDM	NULL
2017-12-12	NULL	UDM	NULL

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

Tricky semantics

How can we evaluate **queries** on Codd tables?

Booking

date	teacher	class	room
2017-11-21	Silviu	UDM	Saphir
2017-11-28	Antoine	UDM	Saphir
2017-12-05	Antoine	UDM	NULL
2017-12-12	NULL	UDM	NULL

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

Booking

date	teacher	class	room
2017-11-21	Silviu	UDM	Saphir

Tricky semantics

How can we evaluate **queries** on Codd tables?

Booking

date	teacher	class	room
2017-11-21	Silviu	UDM	Saphir
2017-11-28	Antoine	UDM	Saphir
2017-12-05	Antoine	UDM	NULL
2017-12-12	NULL	UDM	NULL

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

Booking

date	teacher	class	room
2017-11-21	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
2017-12-05	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
2017-12-05	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
2017-12-05	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
2017-11-21	Silviu	UDM	Saphir
2017-11-28	Antoine	UDM	Jade
2017-12-05	Antoine	UDM	NULL

Three-valued logic (more traps!)

Booking

date	teacher	class	room
2017-11-21	Silviu	UDM	Saphir
2017-11-28	Antoine	UDM	Jade
2017-12-05	Antoine	UDM	NULL

Repairs

room	cause
C42	lavatory leak
NULL	leopard

Three-valued logic (more traps!)

Booking

date	teacher	class	room
2017-11-21	Silviu	UDM	Saphir
2017-11-28	Antoine	UDM	Jade
2017-12-05	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
2017-11-21	Silviu	UDM	Saphir
2017-11-28	Antoine	UDM	Jade
2017-12-05	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				Repairs	
date	teacher	class	room	room	cause
2017-11-21	Silviu	UDM	Saphir	C42	lavatory leak
2017-11-28	Antoine	UDM	Jade	NULL	leopard
2017-12-05	Antoine	UDM	NULL		

```
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				Repairs	
date	teacher	class	room	room	cause
2017-11-21	Silviu	UDM	Saphir	C42	lavatory leak
2017-11-28	Antoine	UDM	Jade	NULL	leopard
2017-12-05	Antoine	UDM	NULL		

```
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 2017-12-05

Three-valued logic (more traps!)

Booking				Repairs	
date	teacher	class	room	room	cause
2017-11-21	Silviu	UDM	Saphir	C42	lavatory leak
2017-11-28	Antoine	UDM	Jade	NULL	leopard
2017-12-05	Antoine	UDM	NULL		

```
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 2017-12-05

```
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
(but with `UNION ALL`, they will)

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
(but with `UNION ALL`, they will)

`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
(but with `UNION ALL`, they will)

`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

SQL

Semantics

V-tables

c-tables

- We fix a **signature** σ :
 - relation **names**
 - associated **arity**
- **Uncertain instance**: each relation name is interpreted by an **uncertain relation**

Uncertain relation

- An **uncertain relation**: set of **possible worlds**,
each possible world is a relation in the usual sense

Uncertain relation

- An **uncertain relation**: set of **possible worlds**,
each possible world is a relation in the usual sense

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

Uncertain relation

- An **uncertain relation**: set of **possible worlds**, each possible world is a relation in the usual sense

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

This is an uncertain relation with **infinitely many** possible worlds, each of which contains **one tuple**

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

U

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**

<u>Booking</u>		<u>Booking</u>
21 S. a		28 a Saphir
<u>Booking</u>		<u>Booking</u>
21 S. b	∪	28 b Saphir
<u>Booking</u>		<u>Booking</u>
21 S. c		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**

<u>Booking</u>		<u>Booking</u>
21 S. a		28 a Saphir
<u>Booking</u>		<u>Booking</u>
21 S. b	∪	28 b Saphir =
<u>Booking</u>		<u>Booking</u>
21 S. c		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**

<table><thead><tr><th colspan="3">Booking</th></tr></thead><tbody><tr><td>21</td><td>S.</td><td>a</td></tr></tbody></table>	Booking			21	S.	a		<table><thead><tr><th colspan="3">Booking</th></tr></thead><tbody><tr><td>28</td><td>a</td><td>Saphir</td></tr></tbody></table>	Booking			28	a	Saphir		<table><thead><tr><th colspan="3">Booking</th></tr></thead><tbody><tr><td>21</td><td>S.</td><td>a</td></tr><tr><td>28</td><td>b</td><td>Saphir</td></tr></tbody></table>	Booking			21	S.	a	28	b	Saphir
Booking																									
21	S.	a																							
Booking																									
28	a	Saphir																							
Booking																									
21	S.	a																							
28	b	Saphir																							
<table><thead><tr><th colspan="3">Booking</th></tr></thead><tbody><tr><td>21</td><td>S.</td><td>b</td></tr></tbody></table>	Booking			21	S.	b	\cup	<table><thead><tr><th colspan="3">Booking</th></tr></thead><tbody><tr><td>28</td><td>b</td><td>Saphir</td></tr></tbody></table>	Booking			28	b	Saphir	$=$	<table><thead><tr><th colspan="3">Booking</th></tr></thead><tbody><tr><td>21</td><td>S.</td><td>b</td></tr><tr><td>28</td><td>a</td><td>Saphir</td></tr></tbody></table>	Booking			21	S.	b	28	a	Saphir
Booking																									
21	S.	b																							
Booking																									
28	b	Saphir																							
Booking																									
21	S.	b																							
28	a	Saphir																							
<table><thead><tr><th colspan="3">Booking</th></tr></thead><tbody><tr><td>21</td><td>S.</td><td>c</td></tr></tbody></table>	Booking			21	S.	c		<table><thead><tr><th colspan="3">Booking</th></tr></thead><tbody><tr><td>28</td><td>c</td><td>Saphir</td></tr></tbody></table>	Booking			28	c	Saphir		<table><thead><tr><th colspan="3">Booking</th></tr></thead><tbody><tr><td>21</td><td>S.</td><td>b</td></tr><tr><td>28</td><td>a</td><td>Saphir</td></tr></tbody></table>	Booking			21	S.	b	28	a	Saphir
Booking																									
21	S.	c																							
Booking																									
28	c	Saphir																							
Booking																									
21	S.	b																							
28	a	Saphir																							
\vdots		\vdots		\vdots																					

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

<table><thead><tr><th colspan="3">Booking</th></tr></thead><tbody><tr><td>21</td><td>S.</td><td>NULL</td></tr></tbody></table>	Booking			21	S.	NULL	stands for	<table><thead><tr><th colspan="3">Booking</th></tr></thead><tbody><tr><td>21</td><td>S.</td><td>a</td></tr></tbody></table>	Booking			21	S.	a
Booking														
21	S.	NULL												
Booking														
21	S.	a												
		<table><thead><tr><th colspan="3">Booking</th></tr></thead><tbody><tr><td>21</td><td>S.</td><td>b</td></tr></tbody></table>	Booking			21	S.	b						
Booking														
21	S.	b												
		<table><thead><tr><th colspan="3">Booking</th></tr></thead><tbody><tr><td>21</td><td>S.</td><td>c</td></tr></tbody></table>	Booking			21	S.	c						
Booking														
21	S.	c												
		⋮												

Back to the silly example

Booking

date	teacher	class	room
2017-12-12	Antoine	UDM	NULL
2018-01-09	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
2017-12-12	Antoine	UDM	NULL
2018-01-09	Silviu	UDM	Sap.

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

→ How to represent the result?

Representing the output

12	A.	UDM	NULL
09	S.	UDM	Sap.

Representing the output

12	A.	UDM	NULL
09	S.	UDM	Sap.

represents

Representing the output

12	A.	UDM	NULL
09	S.	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

12	A.	UDM	NULL
09	S.	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

12	A.	UDM	NULL
09	S.	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

12	A.	UDM	NULL
09	S.	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

12	A.	UDM	NULL
09	S.	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
----	----	-----	------

Uncertain relation : set of possible worlds

Representation system definition

Uncertain relation : set of possible worlds

Uncertainty framework: short way to represent uncertain relations

- Here, **Codd tables**

Representation system definition

Uncertain relation : set of possible worlds

Uncertainty framework: short way to represent uncertain relations

- Here, **Codd tables**

Query language: here, relational algebra

Representation system definition

Uncertain relation : set of possible worlds

Uncertainty framework: short way to represent uncertain relations

- Here, **Codd tables**

Query language: here, relational algebra

Definition (Strong representation system)

For any query in the language,

Representation system definition

Uncertain relation : set of possible worlds

Uncertainty framework: short way to represent uncertain relations

- Here, **Codd tables**

Query language: here, relational algebra

Definition (Strong representation system)

For any query in the language,

on uncertain relations represented in the framework,

Representation system definition

Uncertain relation : set of possible worlds

Uncertainty framework: short way to represent uncertain relations

- Here, **Codd tables**

Query language: here, relational algebra

Definition (Strong representation system)

*For any query in the language,
on uncertain relations represented in the framework,
the uncertain relation obtained by evaluating the query*

Representation system definition

Uncertain relation : set of possible worlds

Uncertainty framework: short way to represent uncertain relations

- Here, **Codd tables**

Query language: here, relational algebra

Definition (Strong representation system)

*For any query in the language,
on uncertain relations represented in the framework,
the uncertain relation obtained by evaluating the query
can also be represented in the framework*

Representation system definition

Uncertain relation : set of possible worlds

Uncertainty framework: short way to represent uncertain relations

- Here, **Codd tables**

Query language: here, relational algebra

Definition (Strong representation system)

*For any query in the language,
on uncertain relations represented in the framework,
the uncertain relation 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
2017-12-05	Antoine	UDM	NULL

Are Codd tables a representation system?

Member		Booking			
id	class	date	teacher	class	room
1	UDM				
2	UDM	2017-12-05	Antoine	UDM	NULL
3	IE				

Can we represent **Member** ⋈ **Booking**?

Are Codd tables a representation system?

Member		Booking			
id	class	date	teacher	class	room
1	UDM				
2	UDM	2017-12-05	Antoine	UDM	NULL
3	IE				

Can we represent **Member** ⋈ **Booking**?

Member ⋈ Booking				
id	date	teacher	class	room
1	2017-12-05	Antoine	UDM	NULL
2	2017-12-05	Antoine	UDM	NULL

Are Codd tables a representation system?

Member		Booking			
id	class	date	teacher	class	room
1	UDM				
2	UDM	2017-12-05	Antoine	UDM	NULL
3	IE				

Can we represent **Member** ⋈ **Booking**?

Member ⋈ Booking				
id	date	teacher	class	room
1	2017-12-05	Antoine	UDM	NULL
2	2017-12-05	Antoine	UDM	NULL

→ Can you spot the **problem**?

Multiple values

- When querying Codd tables, we may **duplicate NULLs**
- We cannot represent that two **NULLs** are the **same**
- This may cause **problems!**

Multiple values example

Booking

date	teacher	class	room
2017-12-12	Antoine	UDM	NULL
2018-01-09	Silviu	UDM	Saphir

$$\Pi_{\text{room}}(\text{Booking}) - \Pi_{\text{room}}(\sigma_{\text{teacher}=\text{"Antoine"}}(\text{Booking}))$$

Multiple values example

Booking

date	teacher	class	room
2017-12-12	Antoine	UDM	NULL
2018-01-09	Silviu	UDM	Saphir

$$\Pi_{\text{room}}(\text{Booking}) - \Pi_{\text{room}}(\sigma_{\text{teacher}=\text{"Antoine"}}(\text{Booking}))$$

According to SQL

According to semantics

Saphir

Multiple values example

Booking

date	teacher	class	room
2017-12-12	Antoine	UDM	NULL
2018-01-09	Silviu	UDM	Saphir

$$\Pi_{\text{room}}(\text{Booking}) - \Pi_{\text{room}}(\sigma_{\text{teacher}=\text{"Antoine"}}(\text{Booking}))$$

According to SQL

According to semantics

Saphir

But if we try to represent **intermediate expressions**?

Multiple values example

Booking

date	teacher	class	room
2017-12-12	Antoine	UDM	NULL
2018-01-09	Silviu	UDM	Saphir

$$\Pi_{\text{room}}(\text{Booking}) - \Pi_{\text{room}}(\sigma_{\text{teacher}=\text{"Antoine"}}(\text{Booking}))$$

According to SQL

According to semantics

Saphir

But if we try to represent intermediate expressions?

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

$\Pi_{\text{room}}(\sigma_{\text{teacher}=\text{"Antoine"}}(\text{Booking}))$

NULL

NULL

Saphir

Table of contents

SQL

Semantics

V-tables

c-tables

v-tables

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

v-tables

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

Member		Booking			
id	class	date	teacher	class	room
1	UDM				
2	UDM	2017-12-05	NULL ₁	UDM	NULL ₂
3	IE				

v-tables

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

Member		Booking			
id	class	date	teacher	class	room
1	UDM				
2	UDM	2017-12-05	NULL ₁	UDM	NULL ₂
3	IE				

Member ⋈ Booking				
id	date	teacher	class	room
1	2017-12-05	NULL ₁	UDM	NULL ₂
2	2017-12-05	NULL ₁	UDM	NULL ₂

v-table semantics

1	2017-12-05	NULL ₁	UDM	NULL ₂
2	2017-12-05	NULL ₁	UDM	NULL ₂

v-table semantics

1	2017-12-05	NULL ₁	UDM	NULL ₂
2	2017-12-05	NULL ₁	UDM	NULL ₂

represents

1	2017-12-05	aa	UDM	bb
2	2017-12-05	aa	UDM	bb

1	2017-12-05	ccc	UDM	ddd
2	2017-12-05	ccc	UDM	ddd

1	2017-12-05	e	UDM	e
2	2017-12-05	e	UDM	e

⋮

Are v-tables a representation system?

Member

id	class
1	UDM
2	UDM
3	IE

Booking

date	teacher	class	room
2017-12-05	Antoine	UDM	NULL ₁

Are v-tables a representation system?

Member		Booking			
id	class	date	teacher	class	room
1	UDM				
2	UDM	2017-12-05	Antoine	UDM	NULL ₁
3	IE				

Can we represent **Member** ⋈ **Booking** as a v-table?

Are v-tables a representation system?

Member		Booking			
id	class	date	teacher	class	room
1	UDM				
2	UDM	2017-12-05	Antoine	UDM	NULL ₁
3	IE				

Can we represent **Member** ⋈ **Booking** as a v-table?

Member ⋈ Booking				
id	date	teacher	class	room
1	2017-12-05	Antoine	UDM	NULL ₁
2	2017-12-05	Antoine	UDM	NULL ₁

Are v-tables a representation system? (2)

Member		Booking			
id	class	date	teacher	class	room
1	UDM				
2	UDM	2017-12-05	NULL ₁	UDM	NULL ₂
3	NULL ₀				

Are v-tables a representation system? (2)

Member		Booking			
id	class	date	teacher	class	room
1	UDM				
2	UDM	2017-12-05	NULL ₁	UDM	NULL ₂
3	NULL ₀				

Member ⋈ Booking					
id	date	teacher	class	room	
1	2017-12-05	NULL ₁	UDM	NULL ₂	
2	2017-12-05	NULL ₁	UDM	NULL ₂	
3	2017-12-05	NULL ₁	UDM	NULL ₂	if NULL ₀ is "UDM"

Problem

- v-tables cannot represent **optional rows**
 - 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

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

id	room	<i>condition</i>
1	NULL ₂	
2	NULL ₂	
3	NULL ₂	if NULL ₀ is "UDM"

Rooms	
room	seats
C42	20
NULL ₃	25

Condition example

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

id	room	condition
1	NULL ₂	
2	NULL ₂	
3	NULL ₂	if NULL ₀ is "UDM"

Rooms

room	seats
C42	20
NULL ₃	25

$R \bowtie \text{Rooms}$

id	room	seats	condition
----	------	-------	-----------

Condition example

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

id	room	condition
1	NULL_2	
2	NULL_2	
3	NULL_2	if NULL_0 is "UDM"

Rooms	
room	seats
C42	20
NULL_3	25

$R \bowtie \text{Rooms}$

id	room	seats	condition
1	NULL_2	20	if NULL_2 is "C42"
1	NULL_2	25	if NULL_2 is NULL_3

Condition example

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

id	room	condition
1	$NULL_2$	
2	$NULL_2$	
3	$NULL_2$	if $NULL_0$ is "UDM"

Rooms	
room	seats
C42	20
$NULL_3$	25

$R \bowtie Rooms$

id	room	seats	condition
1	$NULL_2$	20	if $NULL_2$ is "C42"
1	$NULL_2$	25	if $NULL_2$ is $NULL_3$
2	$NULL_2$	20	if $NULL_2$ is "C42"
2	$NULL_2$	25	if $NULL_2$ is $NULL_3$

Condition example

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

id	room	condition
1	NULL ₂	
2	NULL ₂	
3	NULL ₂	if NULL ₀ is "UDM"

Rooms	
room	seats
C42	20
NULL ₃	25

$R \bowtie \text{Rooms}$

id	room	seats	condition
1	NULL ₂	20	if NULL ₂ is "C42"
1	NULL ₂	25	if NULL ₂ is NULL ₃
2	NULL ₂	20	if NULL ₂ is "C42"
2	NULL ₂	25	if NULL ₂ is NULL ₃
3	NULL ₂	20	if NULL ₂ is "C42" and NULL ₀ is "UDM"
3	NULL ₂	25	if NULL ₂ is NULL ₃ and NULL ₀ is "UDM"

Table of contents

SQL

Semantics

V-tables

c-tables

- Named **NULLs**, plus **conditions** on tuples
- Conditions can use:

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

- Named `NULLs`, plus `conditions` on tuples
- Conditions can use:
 - `true`
 - `false`
 - `NULLi = NULLj`

- Named `NULLs`, plus `conditions` on tuples
- Conditions can use:
 - `true`
 - `false`
 - `NULLi = NULLj`
 - `NULLi = "value"`

- Named **NULLs**, plus **conditions** on tuples
- Conditions can use:
 - **true**
 - **false**
 - $NULL_i = NULL_j$
 - $NULL_i = \text{"value"}$
 - **Boolean** operators

- Named **NULLs**, plus **conditions** on tuples
- Conditions can use:
 - **true**
 - **false**
 - $NULL_i = NULL_j$
 - $NULL_i = \text{"value"}$
 - **Boolean** operators

→ Are **c-tables** a **strong representation system**?

Relational algebra operators: product

S	
s	<i>condition</i>
s_1	C_1
s_2	C_2

Relational algebra operators: product

S		T	
s	<i>condition</i>	t	<i>condition</i>
s_1	C_1	t_1	D_1
s_2	C_2	t_2	D_2

Relational algebra operators: product

S		T	
s	<i>condition</i>	t	<i>condition</i>
s_1	C_1	t_1	D_1
s_2	C_2	t_2	D_2

S × T		
s	t	<i>condition</i>

Relational algebra operators: product

S		T	
s	<i>condition</i>	t	<i>condition</i>
s_1	C_1	t_1	D_1
s_2	C_2	t_2	D_2

S \times T		
s	t	<i>condition</i>
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	<i>condition</i>
s_0	C_0
s_1	C_1

Relational algebra operators: union

S		S_2	
s	<i>condition</i>	s	<i>condition</i>
s_0	C_0	s_0	D_0
s_1	C_1	s_2	D_2

Relational algebra operators: union

S		S_2	
s	<i>condition</i>	s	<i>condition</i>
s_0	C_0	s_0	D_0
s_1	C_1	s_2	D_2

$S \cup S_2$	
s	<i>condition</i>

Relational algebra operators: union

S		S₂	
s	<i>condition</i>	s	<i>condition</i>
s_0	C_0	s_0	D_0
s_1	C_1	s_2	D_2

S \cup S₂	
s	<i>condition</i>
s_0	C_0 or D_0
s_1	C_1
s_2	D_2

Relational algebra operators: project

S		
s	t	<i>condition</i>
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	<i>condition</i>
s_0	t_0	C_0
s_0	t_1	C_1
s_2	t_2	C_2

$\Pi_s(S)$	
s	<i>condition</i>

Relational algebra operators: project

S		
s	t	<i>condition</i>
s_0	t_0	C_0
s_0	t_1	C_1
s_2	t_2	C_2

$\Pi_s(S)$	
s	<i>condition</i>
s_0	C_0 or C_1
s_2	C_2

Relational algebra operators: select (1)

S

s	t	<i>condition</i>
42	t_0	C_0
43	t_1	C_1
NULL;	t_2	C_2

Relational algebra operators: select (1)

S		
s	t	<i>condition</i>
42	t_0	C_0
43	t_1	C_1
NULL;	t_2	C_2

$\sigma_{s="42"}(S)$		
s	t	<i>condition</i>

Relational algebra operators: select (1)

S		
s	t	<i>condition</i>
42	t_0	C_0
43	t_1	C_1
NULL;	t_2	C_2

$\sigma_{s="42"}(S)$		
s	t	<i>condition</i>
42	t_0	C_0

Relational algebra operators: select (1)

S		
s	t	<i>condition</i>
42	t_0	C_0
43	t_1	C_1
NULL _j	t_2	C_2

$\sigma_{s="42"}(S)$		
s	t	<i>condition</i>
42	t_0	C_0
NULL _j	t_2	

Relational algebra operators: select (1)

S

s	t	<i>condition</i>
42	t_0	C_0
43	t_1	C_1
NULL_j	t_2	C_2

$\sigma_{s="42"}(S)$

s	t	<i>condition</i>
42	t_0	C_0
NULL_j	t_2	C_2 and NULL_j = "42"

Relational algebra operators: select (2)

S

s	t	<i>condition</i>
42	42	C_0
43	42	C_1
$NULL_j$	42	C_2
42	$NULL_j$	C_3
$NULL_p$	$NULL_q$	C_4

Relational algebra operators: select (2)

S

s	t	<i>condition</i>
42	42	C_0
43	42	C_1
$NULL_j$	42	C_2
42	$NULL_j$	C_3
$NULL_p$	$NULL_q$	C_4

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

s	t	<i>condition</i>
----------	----------	------------------

Relational algebra operators: select (2)

S

s	t	<i>condition</i>
42	42	C_0
43	42	C_1
$NULL_j$	42	C_2
42	$NULL_j$	C_3
$NULL_p$	$NULL_q$	C_4

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

s	t	<i>condition</i>
42	42	

Relational algebra operators: select (2)

S

s	t	<i>condition</i>
42	42	C_0
43	42	C_1
$NULL_j$	42	C_2
42	$NULL_j$	C_3
$NULL_p$	$NULL_q$	C_4

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

s	t	<i>condition</i>
42	42	C_0

Relational algebra operators: select (2)

S

s	t	<i>condition</i>
42	42	C_0
43	42	C_1
$NULL_j$	42	C_2
42	$NULL_j$	C_3
$NULL_p$	$NULL_q$	C_4

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

s	t	<i>condition</i>
42	42	C_0
$NULL_j$	42	

Relational algebra operators: select (2)

S

s	t	<i>condition</i>
42	42	C_0
43	42	C_1
$NULL_j$	42	C_2
42	$NULL_j$	C_3
$NULL_p$	$NULL_q$	C_4

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

s	t	<i>condition</i>
42	42	C_0
$NULL_j$	42	C_2 and $NULL_j = "42"$

Relational algebra operators: select (2)

S

s	t	<i>condition</i>
42	42	C_0
43	42	C_1
$NULL_j$	42	C_2
42	$NULL_j$	C_3
$NULL_p$	$NULL_q$	C_4

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

s	t	<i>condition</i>
42	42	C_0
$NULL_j$	42	C_2 and $NULL_j = "42"$
42	$NULL_j$	

Relational algebra operators: select (2)

S

s	t	condition
42	42	C_0
43	42	C_1
$NULL_j$	42	C_2
42	$NULL_j$	C_3
$NULL_p$	$NULL_q$	C_4

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

s	t	condition
42	42	C_0
$NULL_j$	42	C_2 and $NULL_j = "42"$
42	$NULL_j$	C_3 and $"42" = NULL_j$

Relational algebra operators: select (2)

S

s	t	condition
42	42	C_0
43	42	C_1
$NULL_j$	42	C_2
42	$NULL_j$	C_3
$NULL_p$	$NULL_q$	C_4

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

s	t	condition
42	42	C_0
$NULL_j$	42	C_2 and $NULL_j = "42"$
42	$NULL_j$	C_3 and $"42" = NULL_j$
$NULL_p$	$NULL_q$	

Relational algebra operators: select (2)

S

s	t	condition
42	42	C_0
43	42	C_1
$NULL_j$	42	C_2
42	$NULL_j$	C_3
$NULL_p$	$NULL_q$	C_4

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

s	t	condition
42	42	C_0
$NULL_j$	42	C_2 and $NULL_j = "42"$
42	$NULL_j$	C_3 and $"42" = NULL_j$
$NULL_p$	$NULL_q$	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**
 - 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**
 - 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 $(NULL_i = "42" \text{ and } NULL_j = "42") \text{ or } ((NULL_k = NULL_j \text{ or } NULL_j = "43") \text{ and } (NULL_i = NULL_j))$

Problems on c-tables

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

Member ⋈ **Booking**

id	date	teacher	class	room	
1	2017-12-05	$NULL_1$	UDM	$NULL_2$	
2	2017-12-05	$NULL_1$	UDM	$NULL_2$	
3	2017-12-05	$NULL_1$	UDM	$NULL_2$	<i>if $NULL_0$ is "UDM"</i>

Problems on c-tables

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

Member ⋈ **Booking**

id	date	teacher	class	room	
1	2017-12-05	$NULL_1$	UDM	$NULL_2$	
2	2017-12-05	$NULL_1$	UDM	$NULL_2$	
3	2017-12-05	$NULL_1$	UDM	$NULL_2$	<i>if $NULL_0$ is "UDM"</i>

What can we **ask** about it?

Problems on c-tables

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

Member ⋈ **Booking**

id	date	teacher	class	room
1	2017-12-05	NULL ₁	UDM	NULL ₂
2	2017-12-05	NULL ₁	UDM	NULL ₂
3	2017-12-05	NULL ₁	UDM	NULL ₂

if NULL₀ is "UDM"

What can we **ask** about it?

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

Problems on c-tables

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

Member ⋈ **Booking**

id	date	teacher	class	room
1	2017-12-05	NULL ₁	UDM	NULL ₂
2	2017-12-05	NULL ₁	UDM	NULL ₂
3	2017-12-05	NULL ₁	UDM	NULL ₂

if NULL₀ is "UDM"

What can we **ask** about it?

- At the **relation** level
 - Is an input **relation** a **possible world**?
 - Is an input **relation** 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 ⋈ **Booking**

id	date	teacher	class	room
1	2017-12-05	NULL ₁	UDM	NULL ₂
2	2017-12-05	NULL ₁	UDM	NULL ₂
3	2017-12-05	NULL ₁	UDM	NULL ₂

if NULL₀ is "UDM"

What can we **ask** about it?

- At the **relation** level
 - Is an input **relation** a **possible world**?
 - Is an input **relation** 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?

→ 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>.