

Uncertain Data Management

NULLs

Antoine Amarilli¹, Silviu Maniu²

¹Télécom ParisTech

²LRI

November 30th, 2015



NULLS

Represent **missing information** in a relation.

NULLs

Represent **missing information** in a relation.

Book

date	teacher	class	room
2015-11-23	Antoine	UDM	C47
2015-11-30	Antoine	UDM	C017
2015-12-07	Antoine	UDM	NULL
2015-12-14	NULL	UDM	NULL

NULLs

Represent **missing information** in a relation.

Book

date	teacher	class	room
2015-11-23	Antoine	UDM	C47
2015-11-30	Antoine	UDM	C017
2015-12-07	Antoine	UDM	NULL
2015-12-14	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

Book

date	teacher	class	room
2015-11-23	Antoine	UDM	C47
2015-11-30	Antoine	UDM	C017
2015-12-07	Antoine	UDM	NULL
2015-12-14	NULL	UDM	NULL

Semantics

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

Book

date	teacher	class	room
2015-11-23	Antoine	UDM	C47
2015-11-30	Antoine	UDM	C017
2015-12-07	Antoine	UDM	C42
2015-12-14	Silviu	UDM	C42

Semantics

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

Book

date	teacher	class	room
2015-11-23	Antoine	UDM	C47
2015-11-30	Antoine	UDM	C017
2015-12-07	Antoine	UDM	xbecz
2015-12-14	gruiiik	UDM	buuuk

Tricky semantics

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

Tricky semantics

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

Book

date	teacher	class	room
2015-11-23	Antoine	UDM	C47
2015-11-30	Antoine	UDM	C017
2015-12-07	Antoine	UDM	NULL
2015-12-14	NULL	UDM	NULL

```
SELECT * FROM Book WHERE room='C47';
```

Tricky semantics

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

Book

date	teacher	class	room
2015-11-23	Antoine	UDM	C47
2015-11-30	Antoine	UDM	C017
2015-12-07	Antoine	UDM	NULL
2015-12-14	NULL	UDM	NULL

```
SELECT * FROM Book WHERE room='C47';
```

Book

date	teacher	class	room
2015-11-23	Antoine	UDM	C47

Tricky semantics

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

Book

date	teacher	class	room
2015-11-23	Antoine	UDM	C47
2015-11-30	Antoine	UDM	C017
2015-12-07	Antoine	UDM	NULL
2015-12-14	NULL	UDM	NULL

```
SELECT * FROM Book WHERE room='C47';
```

Book

date	teacher	class	room
2015-11-23	Antoine	UDM	C47

→ **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 Book 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 Book 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 Book WHERE room=NULL`
→ Returns an empty result
- What does the following do?
`SELECT * FROM Book 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 Book WHERE room=NULL`
→ Returns an empty result
- What does the following do?
`SELECT * FROM Book 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!

Book			
date	teacher	class	room
2015-12-07	Antoine	UDM	NULL

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

Three-valued logic (more complaints)

This is **silly** in terms of semantics!

Book			
date	teacher	class	room
2015-12-07	Antoine	UDM	NULL

SELECT * **FROM** Book **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!

Book			
date	teacher	class	room
2015-12-07	Antoine	UDM	NULL

SELECT * **FROM** Book **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!)

Book

date	teacher	class	room
2015-11-23	Antoine	UDM	C47
2015-11-30	Antoine	UDM	C017
2015-12-03	Antoine	UDM	NULL

Three-valued logic (more traps!)

Book

date	teacher	class	room
2015-11-23	Antoine	UDM	C47
2015-11-30	Antoine	UDM	C017
2015-12-03	Antoine	UDM	NULL

Repairs

room	cause
C42	lavatory leak
NULL	leopard

Three-valued logic (more traps!)

Book

date	teacher	class	room
2015-11-23	Antoine	UDM	C47
2015-11-30	Antoine	UDM	C017
2015-12-03	Antoine	UDM	NULL

Repairs

room	cause
C42	lavatory leak
NULL	leopard

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


Three-valued logic (more traps!)

Book

date	teacher	class	room
2015-11-23	Antoine	UDM	C47
2015-11-30	Antoine	UDM	C017
2015-12-03	Antoine	UDM	NULL

Repairs

room	cause
C42	lavatory leak
NULL	leopard

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

→ Empty result!

Three-valued logic (more traps!)

Book				Repairs	
date	teacher	class	room	room	cause
2015-11-23	Antoine	UDM	C47	C42	lavatory leak
2015-11-30	Antoine	UDM	C017	NULL	leopard
2015-12-03	Antoine	UDM	NULL		

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

→ Empty result!

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

Three-valued logic (more traps!)

Book				Repairs	
date	teacher	class	room	room	cause
2015-11-23	Antoine	UDM	C47	C42	lavatory leak
2015-11-30	Antoine	UDM	C017	NULL	leopard
2015-12-03	Antoine	UDM	NULL		

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

→ Empty result!

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

→ Does **not** contain the **NULL** for 2015-12-03

Three-valued logic (more traps!)

Book				Repairs	
date	teacher	class	room	room	cause
2015-11-23	Antoine	UDM	C47	C42	lavatory leak
2015-11-30	Antoine	UDM	C017	NULL	leopard
2015-12-03	Antoine	UDM	NULL		

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

→ Empty result!

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

→ Does **not** contain the **NULL** for 2015-12-03

```
SELECT * FROM Book 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

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

- 1 SQL
- 2 Semantics**
- 3 V-tables
- 4 c-tables

Semantics

- We fix a **signature** σ :
 - relation **names**
 - 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	...
23	A.	a	23	A.	b	23	A.	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

23	A.	a
----	----	---

Booking

23	A.	b
----	----	---

Booking

23	A.	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

23	A.	a
----	----	---

Booking

23	A.	b
----	----	---

U

Booking

23	A.	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		Booking
23 A. a		30 a C017

Booking	∪	Booking
23 A. b		30 b C017

Booking		Booking
23 A. c		30 c C017

⋮		⋮
---	--	---

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>	
23 A. a		30 a C017	

<u>Booking</u>	∪	<u>Booking</u>	=
23 A. b		30 b C017	

<u>Booking</u>		<u>Booking</u>	
23 A. c		30 c C017	

⋮		⋮	
---	--	---	--

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 style="border-collapse: collapse; margin: auto;"> <tr><th style="border-bottom: 1px solid black; padding: 5px;">Booking</th></tr> <tr><td style="padding: 5px;">23 A. a</td></tr> </table>	Booking	23 A. a		<table style="border-collapse: collapse; margin: auto;"> <tr><th style="border-bottom: 1px solid black; padding: 5px;">Booking</th></tr> <tr><td style="padding: 5px;">30 a C017</td></tr> </table>	Booking	30 a C017		<table style="border-collapse: collapse; margin: auto;"> <tr><th style="border-bottom: 1px solid black; padding: 5px;">Booking</th></tr> <tr><td style="padding: 5px;">23 A. a</td></tr> <tr><td style="padding: 5px;">30 b C017</td></tr> </table>	Booking	23 A. a	30 b C017
Booking											
23 A. a											
Booking											
30 a C017											
Booking											
23 A. a											
30 b C017											
<table style="border-collapse: collapse; margin: auto;"> <tr><th style="border-bottom: 1px solid black; padding: 5px;">Booking</th></tr> <tr><td style="padding: 5px;">23 A. b</td></tr> </table>	Booking	23 A. b	∪	<table style="border-collapse: collapse; margin: auto;"> <tr><th style="border-bottom: 1px solid black; padding: 5px;">Booking</th></tr> <tr><td style="padding: 5px;">30 b C017</td></tr> </table>	Booking	30 b C017	=	<table style="border-collapse: collapse; margin: auto;"> <tr><th style="border-bottom: 1px solid black; padding: 5px;">Booking</th></tr> <tr><td style="padding: 5px;">23 A. b</td></tr> <tr><td style="padding: 5px;">30 a C017</td></tr> </table>	Booking	23 A. b	30 a C017
Booking											
23 A. b											
Booking											
30 b C017											
Booking											
23 A. b											
30 a C017											
<table style="border-collapse: collapse; margin: auto;"> <tr><th style="border-bottom: 1px solid black; padding: 5px;">Booking</th></tr> <tr><td style="padding: 5px;">23 A. c</td></tr> </table>	Booking	23 A. c		<table style="border-collapse: collapse; margin: auto;"> <tr><th style="border-bottom: 1px solid black; padding: 5px;">Booking</th></tr> <tr><td style="padding: 5px;">30 c C017</td></tr> </table>	Booking	30 c C017		<table style="border-collapse: collapse; margin: auto;"> <tr><th style="border-bottom: 1px solid black; padding: 5px;">Booking</th></tr> <tr><td style="padding: 5px;">23 A. b</td></tr> <tr><td style="padding: 5px;">30 a C017</td></tr> </table>	Booking	23 A. b	30 a C017
Booking											
23 A. c											
Booking											
30 c C017											
Booking											
23 A. b											
30 a C017											
⋮		⋮		⋮							

Representation system

Tables with **NULL** are a **representation** of uncertain tables

Representation system

Tables with **NULL** are a **representation** of uncertain tables

Booking

23	A.	NULL
----	----	------

Representation system

Tables with **NULL** are a **representation** of uncertain tables

Booking

23	A.	NULL
----	----	-------------

stands for

Representation system

Tables with **NULL** are a **representation** of uncertain tables

<table style="border-collapse: collapse; margin: 0 auto;"> <tr> <td colspan="3" style="text-align: center; color: red;">Booking</td> </tr> <tr> <td style="padding: 0 10px;">23</td> <td style="padding: 0 10px;">A.</td> <td style="padding: 0 10px; color: green;">NULL</td> </tr> </table>	Booking			23	A.	NULL	stands for	<table style="border-collapse: collapse; margin: 0 auto;"> <tr> <td colspan="3" style="text-align: center; color: red;">Booking</td> </tr> <tr> <td style="padding: 0 10px;">23</td> <td style="padding: 0 10px;">A.</td> <td style="padding: 0 10px; color: green;">a</td> </tr> </table>	Booking			23	A.	a
Booking														
23	A.	NULL												
Booking														
23	A.	a												
		<table style="border-collapse: collapse; margin: 0 auto;"> <tr> <td colspan="3" style="text-align: center; color: red;">Booking</td> </tr> <tr> <td style="padding: 0 10px;">23</td> <td style="padding: 0 10px;">A.</td> <td style="padding: 0 10px; color: green;">b</td> </tr> </table>	Booking			23	A.	b						
Booking														
23	A.	b												
		<table style="border-collapse: collapse; margin: 0 auto;"> <tr> <td colspan="3" style="text-align: center; color: red;">Booking</td> </tr> <tr> <td style="padding: 0 10px;">23</td> <td style="padding: 0 10px;">A.</td> <td style="padding: 0 10px; color: green;">c</td> </tr> </table>	Booking			23	A.	c						
Booking														
23	A.	c												
		⋮												

Back to the silly example

Book

date	teacher	class	room
2015-12-07	Antoine	UDM	NULL
2015-12-14	Silviu	UDM	C47

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

Back to the silly example

Book

date	teacher	class	room
2015-12-07	Antoine	UDM	NULL
2015-12-14	Silviu	UDM	C47

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

→ How to **represent** the result?

Representing the output

07	A.	UDM	NULL
14	S.	UDM	C47

Representing the output

07	A.	UDM	NULL
14	S.	UDM	C47

represents

Representing the output

07	A.	UDM	NULL
14	S.	UDM	C47

represents

07	A.	UDM	a	07	A.	UDM	b	07	A.	UDM	C42	...
14	S.	UDM	C47	14	S.	UDM	C47	14	S.	UDM	C47	

Representing the output

07	A.	UDM	NULL
14	S.	UDM	C47

represents

07	A.	UDM	a	07	A.	UDM	b	07	A.	UDM	C42	...
14	S.	UDM	C47	14	S.	UDM	C47	14	S.	UDM	C47	

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

Representing the output

07	A.	UDM	NULL
14	S.	UDM	C47

represents

07	A.	UDM	a	07	A.	UDM	b	07	A.	UDM	C42	...
14	S.	UDM	C47	14	S.	UDM	C47	14	S.	UDM	C47	

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

07	A.	UDM	a	07	A.	UDM	b	07	A.	UDM	C42	...
----	----	-----	---	----	----	-----	---	----	----	-----	-----	-----

Representing the output

07	A.	UDM	NULL
14	S.	UDM	C47

represents

07	A.	UDM	a	07	A.	UDM	b	07	A.	UDM	C42	...
14	S.	UDM	C47	14	S.	UDM	C47	14	S.	UDM	C47	

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

07	A.	UDM	a	07	A.	UDM	b	07	A.	UDM	C42	...
----	----	-----	---	----	----	-----	---	----	----	-----	-----	-----

represented as

Representing the output

07	A.	UDM	NULL
14	S.	UDM	C47

represents

07	A.	UDM	a	07	A.	UDM	b	07	A.	UDM	C42	...
14	S.	UDM	C47	14	S.	UDM	C47	14	S.	UDM	C47	

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

07	A.	UDM	a	07	A.	UDM	b	07	A.	UDM	C42	...
----	----	-----	---	----	----	-----	---	----	----	-----	-----	-----

represented as

07	A.	UDM	NULL
----	----	-----	------

Representation system definition

Uncertain instance: set of possible worlds

Representation system definition

Uncertain instance: set of possible worlds

Uncertainty framework: concise way to represent uncertain instances

- Here, **Codd tables**

Representation system definition

Uncertain instance: set of possible worlds

Uncertainty framework: concise way to represent uncertain instances

- Here, **Codd tables**

Query language: here, relational algebra

Representation system definition

Uncertain instance: set of possible worlds

Uncertainty framework: concise 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: concise 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: concise 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: concise 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: concise 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
2015-12-07	Antoine	UDM	NULL

Are Codd tables a representation system?

Member		Booking			
id	class	date	teacher	class	room
1	UDM				
2	UDM	2015-12-07	Antoine	UDM	NULL
3	IE				

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

Are Codd tables a representation system?

Member	
id	class
1	UDM
2	UDM
3	IE

Booking			
date	teacher	class	room
2015-12-07	Antoine	UDM	NULL

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

Member ⋈ Booking				
id	date	teacher	class	room
1	2015-12-07	Antoine	UDM	NULL
2	2015-12-07	Antoine	UDM	NULL

Are Codd tables a representation system?

Member		Booking			
id	class	date	teacher	class	room
1	UDM				
2	UDM	2015-12-07	Antoine	UDM	NULL
3	IE				

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

Member ⋈ Booking				
id	date	teacher	class	room
1	2015-12-07	Antoine	UDM	NULL
2	2015-12-07	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
2015-12-07	Antoine	UDM	NULL
2015-12-14	Silviu	UDM	C47

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

Multiple values example

Booking			
date	teacher	class	room
2015-12-07	Antoine	UDM	NULL
2015-12-14	Silviu	UDM	C47

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

According to SQL

According to semantics

C47

Multiple values example

Booking			
date	teacher	class	room
2015-12-07	Antoine	UDM	NULL
2015-12-14	Silviu	UDM	C47

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

According to SQL

According to semantics

C47

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

Multiple values example

Booking			
date	teacher	class	room
2015-12-07	Antoine	UDM	NULL
2015-12-14	Silviu	UDM	C47

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

According to SQL

According to semantics

C47

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

C47

Table of contents

- 1 SQL
- 2 Semantics
- 3 V-tables**
- 4 c-tables

v-tables

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

v-tables

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

Member		Booking			
id	class	date	teacher	class	room
1	UDM				
2	UDM	2015-12-07	NULL ₁	UDM	NULL ₂
3	IE				

v-tables

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

Member		Booking			
id	class	date	teacher	class	room
1	UDM				
2	UDM	2015-12-07	NULL ₁	UDM	NULL ₂
3	IE				

Member ⋈ Booking					
id	date	teacher	class	room	
1	2015-12-07	NULL ₁	UDM	NULL ₂	
2	2015-12-07	NULL ₁	UDM	NULL ₂	

v-table semantics

1	2015-12-07	NULL ₁	UDM	NULL ₂
2	2015-12-07	NULL ₁	UDM	NULL ₂

v-table semantics

1	2015-12-07	NULL ₁	UDM	NULL ₂
2	2015-12-07	NULL ₁	UDM	NULL ₂
1	2015-12-07	aa	UDM	bb
2	2015-12-07	aa	UDM	bb

v-table semantics

1	2015-12-07	NULL ₁	UDM	NULL ₂
2	2015-12-07	NULL ₁	UDM	NULL ₂
1	2015-12-07	aa	UDM	bb
2	2015-12-07	aa	UDM	bb
1	2015-12-07	ccc	UDM	ddd
2	2015-12-07	ccc	UDM	ddd

v-table semantics

1	2015-12-07	NULL _{L1}	UDM	NULL _{L2}
2	2015-12-07	NULL _{L1}	UDM	NULL _{L2}
1	2015-12-07	aa	UDM	bb
2	2015-12-07	aa	UDM	bb
1	2015-12-07	ccc	UDM	ddd
2	2015-12-07	ccc	UDM	ddd
1	2015-12-07	e	UDM	e
2	2015-12-07	e	UDM	e

Are v-tables a representation system?

Member

id	class
1	UDM
2	UDM
3	NULL ₀

Booking

date	teacher	class	room
2015-12-07	NULL ₁	UDM	NULL ₂

Are v-tables a representation system?

Member

id	class
1	UDM
2	UDM
3	NULL ₀

Booking

date	teacher	class	room
2015-12-07	NULL ₁	UDM	NULL ₂

Member ⋈ Booking

id	date	teacher	class	room
1	2015-12-07	NULL ₁	UDM	NULL ₂
2	2015-12-07	NULL ₁	UDM	NULL ₂
3	2015-12-07	NULL ₁	UDM	NULL ₂

if NULL₀ 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

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

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

Condition example

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

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

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

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 $NULL_2 = \text{"C42"}$
1	$NULL_2$	25	if $NULL_2 = NULL_3$

Condition example

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

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 $NULL_2 = \text{"C42"}$
1	$NULL_2$	25	if $NULL_2 = NULL_3$
2	$NULL_2$	20	if $NULL_2 = \text{"C42"}$
2	$NULL_2$	25	if $NULL_2 = NULL_3$

Condition example

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

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 $NULL_2 = \text{"C42"}$
1	$NULL_2$	25	if $NULL_2 = NULL_3$
2	$NULL_2$	20	if $NULL_2 = \text{"C42"}$
2	$NULL_2$	25	if $NULL_2 = NULL_3$
3	$NULL_2$	20	if $NULL_2 = \text{"C42"}$ and $NULL_0 = \text{"UDM"}$
3	$NULL_2$	25	if $NULL_2 = NULL_3$ and $NULL_0 = \text{"UDM"}$

Table of contents

- 1 SQL
- 2 Semantics
- 3 V-tables
- 4 c-tables**

c-tables

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

c-tables

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

c-tables

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

c-tables

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

c-tables

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

c-tables

- Named **NULLs**, plus **conditions** on tuples
- Conditions can use:
 - **true**
 - **false**
 - $\text{NULL}_i = \text{NULL}_j$
 - $\text{NULL}_i = \text{"value"}$
 - **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 × 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		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

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 ∪ S2	
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 _{<i>j</i>}	t_2	C_2

Relational algebra operators: select (1)

S

s	t	condition
42	t_0	C_0
43	t_1	C_1
NULL _j	t_2	C_2

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

s	t	condition
---	---	-----------

Relational algebra operators: select (1)

S

s	t	condition
42	t_0	C_0
43	t_1	C_1
NULL _j	t_2	C_2

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

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
NULL _j	t_2	C_2

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

s	t	condition
42	t_0	C_0
NULL _j	t_2	

Relational algebra operators: select (1)

S

s	t	condition
42	t_0	C_0
43	t_1	C_1
$NULL_j$	t_2	C_2

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

s	t	condition
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	condition
42	42	C_0
43	42	C_1
$NULL_i$	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
$NULL_i$	42	C_2 and $NULL_i = "42"$

Relational algebra operators: select (2)

S

s	t	condition
42	42	C_0
43	42	C_1
$NULL_i$	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
$NULL_i$	42	C_2 and $NULL_i = "42"$
42	$NULL_j$	

Relational algebra operators: select (2)

S

s	t	condition
42	42	C_0
43	42	C_1
$NULL_i$	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
$NULL_i$	42	C_2 and $NULL_i = "42"$
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
$NULL_i$	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
$NULL_i$	42	C_2 and $NULL_i = "42"$
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
$NULL_i$	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
$NULL_i$	42	C_2 and $NULL_i = "42"$
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 $(NULL_i = \text{"42"} \text{ and } NULL_j = \text{"42"}) \text{ or } ((NULL_k = NULL_j \text{ or } NULL_j = \text{"43"}) \text{ and } (NULL_i = NULL_j))$

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	2015-12-07	NULL ₁	UDM	NULL ₂
2	2015-12-07	NULL ₁	UDM	NULL ₂
3	2015-12-07	NULL ₁	UDM	NULL ₂ if NULL ₀ is "UDM"

Problems on c-tables

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

Member ⋈ **Booking**

id	date	teacher	class	room
1	2015-12-07	NULL ₁	UDM	NULL ₂
2	2015-12-07	NULL ₁	UDM	NULL ₂
3	2015-12-07	NULL ₁	UDM	NULL ₂ <i>if NULL₀ 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	2015-12-07	NULL ₁	UDM	NULL ₂
2	2015-12-07	NULL ₁	UDM	NULL ₂
3	2015-12-07	NULL ₁	UDM	NULL ₂ <i>if NULL₀ is "UDM"</i>

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 ⋈ **Booking**

id	date	teacher	class	room
1	2015-12-07	NULL ₁	UDM	NULL ₂
2	2015-12-07	NULL ₁	UDM	NULL ₂
3	2015-12-07	NULL ₁	UDM	NULL ₂ <i>if NULL₀ is "UDM"</i>

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	2015-12-07	NULL ₁	UDM	NULL ₂	
2	2015-12-07	NULL ₁	UDM	NULL ₂	
3	2015-12-07	NULL ₁	UDM	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?

→ 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>.