Data Wrangling : Project

10^{10^6} Worlds and Beyond : Efcient Representation and Processing of Incomplete Information

Théo Delemazure

Ecole normale supérieure, PSL University

March 8, 2020

How to represents possible world ?

In class, we discussed how to represent a set of possible worlds, using for instance TID, BID and pc-table when these worlds are associated to a probability. In our case, we just look at sets of possible worlds, but not their probabilities.

$\Rightarrow$ How to represent and store an uncertain databases efficiently ?

Motivation

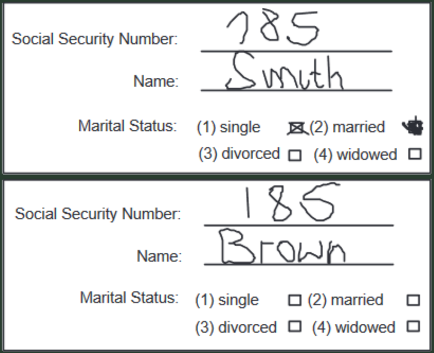


FIGURE – Example of manually completed forms with various possible interpretations

c-table

a	b	c	
0	2	0	$\neg x_1$
0	1	0	$x_1 \vee x_2$
1	4	3	x_2

a	b	c	
0	x_1	0	$x_1 \neq 1$
0	1	x_2	$x_1 = x_2$
1	4	3	$x_3 \neq 0$

TABLE – Examples of c-table according to 2 different definitions

Strong representation...

...but very inefficient in practice

c-table

a	b	c	
0	2	0	$\neg x_1$
0	1	0	$x_1 \vee x_2$
1	4	3	x_2

a	b	c	
0	x_1	0	$x_1 \neq 1$
0	1	x_2	$x_1 = x_2$
1	4	3	$x_3 \neq 0$

TABLE – Examples of c-table according to 2 different definitions

Strong representation...
...but very inefficient in practice

Or-Set relations

id	SSN	Name	Status
0	$\{185, 785\}$	Smith	{S, M}
1	$\{185, 186\}$	Brown	{S, M,D,W}

TABLE – Example of an or-set relation

Easy to represent

...but not a strong representation system : how to represent the fact that the SSN must be unique ?

Or-Set relations

id	SSN	Name	Status
0	$\{185, 785\}$	Smith	{S, M}
1	$\{185, 186\}$	Brown	{S, M,D,W}

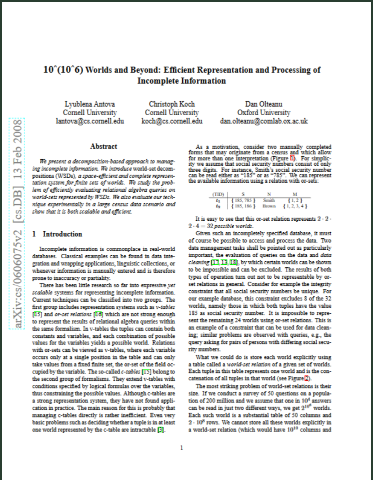
TABLE – Example of an or-set relation

Easy to represent

...but not a strong representation system : how to represent the fact that the SSN must be unique ?

10^{10^6} Worlds and Beyond : Efficient Representation and Processing of Incomplete Information

Authors	*Lyublena Antova, Christoph Koch, and Dan Olteanu*
Year	*2007*
Journal	*IEEE International Conference on Data Engineering*
Citations	197 to this day



10^(10^6) Worlds and Beyond: Efficient Representation and Processing of Incomplete Information

Lyublena Antova, Cornell University, lantova@cs.cornell.edu
Christoph Koch, Cornell University, koch@cs.cornell.edu
Dan Olteanu, Oxford University, dan.olteanu@comlab.ox.ac.uk

Abstract

We present a decomposition-based approach to managing incomplete information. We introduce world-set decompositions (WSDs), *a space-efficient and complete representation system for finite sets of worlds. We study the problem of efficiently evaluating relational algebra queries on world-sets represented by WSDs. We also evaluate our technique experimentally in a large census data scenario and show that it is both scalable and efficient.*

1 Introduction

Incomplete information is commonplace in real-world databases. Classical examples can be found in data integration and wrapping applications, linguistic collections, or whenever information is manually entered and is therefore prone to inaccuracy or partiality.

There has been little research so far into expressive *yet scalable* systems for representing incomplete information. Current techniques can be classified into two groups. The first group includes representation systems such as *v-tables* [15] and *or-set relations* [16] which are not strong enough to represent the results of relational algebra queries within the same formalism. In v-tables the tuples can contain both constants and variables, and each combination of possible values for the variables yields a possible world. Relations with or-sets can be viewed as v-tables, where each variable occurs only at a single position in the table and can only take values from a fixed finite set, the or-set of the field occupied by the variable. The so-called *c-tables* [15] belong to the second group of formalisms. They extend v-tables with conditions specified by logical formulas over the variables, thus constraining the possible values. Although c-tables are a strong representation system, they have not found application in practice. The main reason for this is probably that managing c-tables directly is rather inefficient. Even very basic problems such as deciding whether a tuple is in at least one world represented by the c-table are intractable [3].

As a motivation, consider two manually completed forms that may originate from a census and which allow for more than one interpretation (Figure 1). For simplicity we assume that social security numbers consist of only three digits. For instance, Smith's social security number can be read either as "185" or as "785". We can represent the available information using a relation with or-sets:

(TID)	S	N	M
t_1	{ 185, 785 }	Smith	{ 1, 2 }
t_2	{ 185, 186 }	Brown	{ 1, 2, 3, 4 }

It is easy to see that this or-set relation represents $2 \cdot 2 \cdot 2 \cdot 4 = 32$ *possible worlds*.

Given such an incompletely specified database, it must of course be possible to access and process the data. Two data management tasks shall be pointed out as particularly important, the evaluation of queries on the data and *data cleaning* [17, 13, 18], by which certain worlds can be shown to be impossible and can be excluded. The results of both types of operation turn out not to be representable by or-set relations in general. Consider for example the integrity constraint that all social security numbers be unique. For our example database, this constraint excludes 8 of the 32 worlds, namely those in which both tuples have the value 185 as social security number. It is impossible to represent the remaining 24 worlds using or-set relations. This is an example of a constraint that can be used for data cleaning; similar problems are observed with queries, e.g., the query asking for pairs of persons with differing social security numbers.

What we could do is store each world explicitly using a table called a *world-set relation* of a given set of worlds. Each tuple in this table represents one world and is the concatenation of all tuples in that world (see Figure 2).

The most striking problem of world-set relations is their size. If we conduct a survey of 50 questions on a population of 200 million and we assume that one in 10^4 answers can be read in just two different ways, we get 2^{10^6} worlds. Each such world is a substantial table of 50 columns and $2 \cdot 10^8$ rows. We cannot store all these worlds explicitly in a world-set relation (which would have 10^{10} columns and

1

Summary

Summary

What if we stored every world ?

The idea is to have a relation in which each row is a possible world :

id	t1.SSN	t1.Name	t1.Status	t2.SSN	t2.Name	t2.Status
0	185	Smith	S	186	Brown	S
1	185	Smith	S	186	Brown	M
2	785	Smith	M	185	Brown	S
...	...	...	...	...	...	...
27	785	Smith	M	186	Brown	W

⇒ In most case, there is too much possible world !

What if we stored every world ?

The idea is to have a relation in which each row is a possible world :

id	t1.SSN	t1.Name	t1.Status	t2.SSN	t2.Name	t2.Status
0	185	Smith	S	186	Brown	S
1	185	Smith	S	186	Brown	M
2	785	Smith	M	185	Brown	S
...	...	...	...	...	...	...
27	785	Smith	M	186	Brown	W

$\Rightarrow$ In most case, there is too much possible world !

Example

Many Multi-valued Dependencies that can be avoided.
This lead us to the World-Set Decomposition model :

t_1.SSN	t_2.SSN
185	186
785	186
785	185

×

t_1.Name
Smith

×

t_1.St
S
M

×

t_2.Name
Brown

×

t_2.St
S
M
D
W

Components with one tuple take too much space...

Example

Many Multi-valued Dependencies that can be avoided.
This lead us to the World-Set Decomposition model :

t_1.SSN	t_2.SSN
185	186
785	186
785	185

×

t_1.Name
Smith

×

t_1.St
S
M

×

t_2.Name
Brown

×

t_2.St
S
M
D
W

Components with one tuple take too much space...

Example

This lead us to the World-Set Decomposition with Template relation model :

id	SSN	Name	St
t_1	?	Smith	?
t_2	?	Brown	?

t_1.SSN	t_2.SSN
185	186
785	186
785	185

$\times$

t_1.St
S
M

$\times$

t_2.St
S
M
D
W

WSD : Formal Definition

I present the model for a single relation but this works for any number of relation in the database.

We denote $|R|_{max}$ the maximum number of tuples in the table R

$$|R|_{max} = \max\{|R^W| | W \in \mathcal{W}\}$$

Then, a world-set relation of a world-set $\mathcal{W}$ has the following schema :

$$\{t_i.A_j | i \in [0, |R|_{max}], j \in schema(R)\}$$

And the world $W = (t_1, ..., t_n)$ with $n \leq |R|_{max}$ is represented by the tuple

$$t_W = t_1 \circ t_2 \circ ... \circ t_n \circ \underbrace{(\bot, \cdots, \bot)}_{arity(R)\times(|R|_{max}-n)}$$

Reminder : Strong representation system

Definition

A representation model M for uncertain databases is a strong representation model for a query language $\mathcal{Q}$, if for every set of possible world $\mathcal{W}$ and every query $q \in \mathcal{Q}$, we have that $Q(\mathcal{W})$ can be represented in M.

Properties of World Set decomposition

Definition

Let $\mathcal{W}$ be a world-set and $R_{\mathcal{W}}$ a world-set relation representing $\mathcal{W}$. Then a world-set m-decomposition (m-WSD) of $\mathcal{W}$ is a product m-decomposition of $R_{\mathcal{W}}$.

Theorem

Any finite set of possible worlds can be represented as a world-set relation and as a 1-WSD.

Corollary

WSDs are a strong representation system for any relational query language.

Extensions of WSD

If we add a template relation as in the example, the obtained model WSDT remains a strong representation system for any relational query language (since any WSD can be represented as a WSDT and reciprocally)

Moreover, even if we attach a probability to each possible world in the World Set Decomposition, it remains a strong representation system.

How to represent a WSDT in practice ?

We cannot create infinitely many relation in a database, so we need to store information about all clusters in a finite set of relation :

R^0	SSN	Name	St
t_1	?	Smith	S
t_2	?	Brown	?

F	tid	attr	cluster
	t_1	SSN	c_1
	t_2	SSN	c_1
	t_2	St.	c_2

C	tid	attr	lwid	val
	t_1	SSN	0	185
	t_2	SSN	0	186
	t_1	SSN	1	785
	t_2	SSN	1	186
	t_1	SSN	2	785
	t_2	SSN	2	185
	t_2	St.	0	S
	t_2	St.	1	M

Summary

Tools

⇒ Every algorithm presented here is implemented as a `MySQL` query plan managed with the `MySQL` library in `Python`.

Renaming

$$name_1 \rightarrow name_2$$

1. Rename attribute $name_1$ as $name_2$ in R^0.
2. Replace every occurrence of $name_1$ by $name_2$ in C and F.

Selection $A\theta c$

$$height \leq 10$$

1. Add into R^0_{new} every tuple from R^0 which validate the selection condition in at least one world. If the value is NULL in R^0, we must look into the table C.
2. Copy rows from C and F into C_{new} and F_{new} which corresponds to a tuple in R^0 verifying the condition.
3. In the table C_{new}, if the $attribute$ is A and the $value$ does not verify the condition, replace it by NULL.
4. Propagate NULLs.

Selection $A\theta B$

$$height \leq circumference$$

This is almost the same algorithm, but...

1. There is more case to consider when selecting tuple which verify the condition in at least one possible world.
2. We need to merge components referring to the same tuple of R^0, one with the attribute A and the other the attribute B. (The merging is explained later).

$\cdots \times$

t_1.height
100
50

$\times$

t_1.circumference
150
60

$\times \cdots$

Projection

SELECT $height, circumference$

$\Rightarrow$ We don't want to lose information, se we need to merge some components (see example below). The authors suggest to merge all components referring to the same tuple of R^0.

However, we do not need to merge the components of **every** tuples. In particular, if a tuple never occurs in a component with another tuple, then no merging is necessary.

t_1.A
0

×

t_2.A
1

×

t_1.B	t_2.B
$\perp$	1
2	$\perp$

t_1.A	t_2.A
$\perp$	1
0	$\perp$

The merging problem : What is merging ?

tid	att.	lwid	value
...	...	...	...
t_1	$height$	0	5
t_1	$height$	1	3
t_1	$height$	2	1
...	...	...	...
t_1	$circ$	0	1
t_1	$circ$	1	2
...	...	...	...

$\Longrightarrow$

tid	att.	lwid	value
...	...	...	...
t_1	$height$	0	5
t_1	$height$	1	5
t_1	$height$	2	3
t_1	$height$	3	3
t_1	$height$	4	1
t_1	$height$	5	1
...	...	...	...
t_1	$circ$	0	1
t_1	$circ$	1	2
t_1	$circ$	2	1
t_1	$circ$	3	2
t_1	$circ$	4	1
t_1	$circ$	5	2
...	...	...	...

The merging problem : How to merge ?

- Merge two by two : Bad idea, would take too much time (*16 min* vs *50* sec)
- Merge all components at the same time : Bad idea, what if we need to merge 5 components together ?
- Merge components step by step : Only merge a subset of components such that every component is merged only once each step.

The merging problem : How to merge ?

- Merge two by two : Bad idea, would take too much time (*16 min* vs *50* sec)
- Merge all components at the same time : Bad idea, what if we need to merge 5 components together ?
- Merge components step by step : Only merge a subset of components such that every component is merged only once each step.

The merging problem : How to merge ?

- Merge two by two : Bad idea, would take too much time (*16 min* vs *50* sec)
- Merge all components at the same time : Bad idea, what if we need to merge 5 components together ?
- Merge components step by step : Only merge a subset of components such that every component is merged only once each step.

Cross product, Union and Difference

$$R_1 \theta R_2 \text{ with } \theta \in \{\times, \cup, -\}$$

- Cross-product : Need to change the ids of tuples and components and copy each component of R_1, $|R_2|$ times and each component of R_2, $|R_1|$ times.
- Union : Just need to change the ids of tuples and components.
- Difference : Not implemented.

Cross product, Union and Difference

$$R_1 \theta R_2 \text{ with } \theta \in \{\times, \cup, -\}$$

- Cross-product : Need to change the ids of tuples and components and copy each component of R_1, $|R_2|$ times and each component of R_2, $|R_1|$ times.
- Union : Just need to change the ids of tuples and components.
- Difference : Not implemented.

Cross product, Union and Difference

$$R_1 \theta R_2 \text{ with } \theta \in \{\times, \cup, -\}$$

- Cross-product : Need to change the ids of tuples and components and copy each component of R_1, $|R_2|$ times and each component of R_2, $|R_1|$ times.
- Union : Just need to change the ids of tuples and components.
- Difference : Not implemented.

Simple functions

$$height \times 100 \text{ as } height\_cm$$

- Apply the function to values in R^0 which are not NULLL.
- Apply the function to values in C.

Aggregation : count

`count`

- `count` tuples which are in every possible world (they do not appear in F) and put the results in component $0*$.
- Merge every components referring to the same tuple of R^0.
- For each component, select the distinct possible `count` over all the possible worlds.
- Add values of components with one possible world to the component $0*$.

comp.	count
$0*$	421
2	0
2	1
5	0
5	2
5	3

Aggregation : sum

sum

- sum over tuples which are in every possible world (they do not appear in F) and put the results in component $0*$.
- Merge every components referring to the same tuple of R^0.
- For each component, select the distinct possible sum over all the possible worlds.
- Add values of components with one possible world to the component $0*$.

comp.	sum_{height}
$0*$	45682
2	0
2	12
5	0
5	23
5	44

Aggregation : average

avg

- avg over tuples which are in every possible world (they do not appear in F) and put the results in component $0*$.
- Merge every components referring to the same tuple of R^0.
- For each component, select the distinct possible avg over all the possible worlds.
- Add values of components with one possible world to the component $0*$.

comp.	c	avg_{height}
$0*$	421	11.1
2	0	0
2	1	12
5	0	0
5	2	11.5
5	3	14.6

Normalization

⇒ Goal : reduce the size of C.

1. Removing duplicates empty world. After that, we need to reindex the `lwid` of components in which we deleted some worlds.
2. For fields with one possibility in C, insert it in the template relation R^0.

tid	att.	lwid	value
t_1	*height*	0	NULL
t_1	*height*	1	NULL
t_1	*height*	2	1
t_1	*circ*	0	NULL
t_1	*circ*	1	NULL
t_1	*circ*	2	2
t_2	*height*	0	5
t_2	*height*	1	5

$\Longrightarrow$

tid	att.	lwid	value
t_1	*height*	0	NULL
t_1	*height*	1	1
t_1	*circ*	0	NULL
t_1	*circ*	1	2

Normalization : More

A full normalization requires more work...

- Delete every duplicate worlds, not only empty ones.
- Decompose components which can be decomposed.

Summary

IPUMS 5%

Integrated Public Use Microdata Series, 1990

- Description : The publicly available 5% extract from the 1990 US census, consisting of 50 multiple-choice questions.
- Number of tuples : ~ 12.5 Millions.
- Number of attributes : 50 (marital status, american citizenship, english level, fertility, military status, WWII status, etc.).

Adding uncertainty

To add uncertainty, they created artificial noise by replacing some values in the table by multiple possibilities.

For instance, MARITAL STATUS : S $\rightarrow$ { S,M,W }

Density	0.005%	0.01%	0.05%	0.1%
#Components	31117	62331	312730	624449

Searching for a dataset

For my own experiment, I needed a dataset that...

1. Is large enough to test our queries when there is a lot of data
2. Contains natural uncertainty

Searching for a dataset

For my own experiment, I needed a dataset that...

1. Is large enough to test our queries when there is a lot of data
2. Contains natural uncertainty

Searching for a dataset

For my own experiment, I needed a dataset that...

1. Is large enough to test our queries when there is a lot of data
2. Contains natural uncertainty

Tree dataset

- Number of rows : 205 219
- Selected attributes : tree species, height, circumference, stage of development, noticeability,borough, etc.
- Removed attributes : Geographic position and precise location

FIGURE – The biggest tree in Paris

Tree dataset : Uncertainty

- Apparently, 135 tree in Paris are taller than 100 meters ! There is a Tilleul in Bois de Vincennes, which is 881 km high.
- Similarly, there is some trees with a circumference bigger than 1km in Paris !
- More than 15% of trees are 0 meters tall and 11% have a circumference of 0 centimeters.
- Finally, for 28% of the tuple, the stage of development (J, JA, A or M) is not specified.

Tree dataset : Uncertainty

- Apparently, 135 tree in Paris are taller than 100 meters ! There is a Tilleul in Bois de Vincennes, which is 881 km high.
- Similarly, there is some trees with a circumference bigger than 1km in Paris !
- More than 15% of trees are 0 meters tall and 11% have a circumference of 0 centimeters.
- Finally, for 28% of the tuple, the stage of development (J, JA, A or M) is not specified.

Tree dataset : Uncertainty

- Apparently, 135 tree in Paris are taller than 100 meters ! There is a Tilleul in Bois de Vincennes, which is 881 km high.
- Similarly, there is some trees with a circumference bigger than 1km in Paris !
- More than 15% of trees are 0 meters tall and 11% have a circumference of 0 centimeters.
- Finally, for 28% of the tuple, the stage of development (J, JA, A or M) is not specified.

Tree dataset : Uncertainty

- Apparently, 135 tree in Paris are taller than 100 meters ! There is a Tilleul in Bois de Vincennes, which is 881 km high.
- Similarly, there is some trees with a circumference bigger than 1km in Paris !
- More than 15% of trees are 0 meters tall and 11% have a circumference of 0 centimeters.
- Finally, for 28% of the tuple, the stage of development (J, JA, A or M) is not specified.

Tree dataset : Uncertainty

- Apparently, 135 tree in Paris are taller than 100 meters ! There is a Tilleul in Bois de Vincennes, which is 881 km high.
- Similarly, there is some trees with a circumference bigger than 1km in Paris !
- More than 15% of trees are 0 meters tall and 11% have a circumference of 0 centimeters.
- Finally, for 28% of the tuple, the stage of development (J, JA, A or M) is not specified.

Tree dataset : Uncertainty

Here are some rules I applied ;

1	height	If $= 0$, select 3 different heights at random using a gaussian distribution.
2	height	If > 45, add every combination of 1 or 2 consecutive digit giving a value ≤ 45 and if < 100, try to replace the first digit by 1.
3	height important	If $\in [30, 45]$, and important = False, add one world with the same height and important = True and a add worlds by replacing the first digit with 0, 1 or 2.
...	...	...
6	stage of dev.	If not specified, add a world for each possibility.

Tree dataset : Uncertainty

This gives us more than

$$10^{10^4}$$

possible worlds

Adding constraints : The chase

We want to enforce logical constraints.
For instance, the rule *"People who participated in the second world war to have completed their military service"* is represented by :

$$\text{WWII} = 1 \Rightarrow \textsc{military} \neq 4$$

To enforce that, they use an adaptation of the chase technique. The idea is to merge components of the two attributes and remove inconsistent tuples.

Constraints on the Tree Dataset

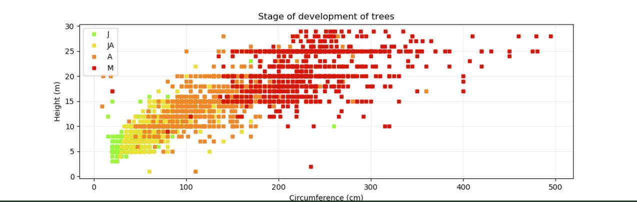


FIGURE – Distribution of height and circumference of trees with different stage of development

	Rule			#Merging	Time (s)
1	$development = "JA"$	$\Rightarrow$	$height \leq 25$	147	31.5
2	$development = "J"$	$\Rightarrow$	$height \leq 20$	1718	93.8
3	$development = "JA"$	$\Rightarrow$	$circ \leq 250$	255	36.4
4	$development = "J"$	$\Rightarrow$	$circ \leq 200$	2036	102.7

Summary

Experiments : Selection and Projection

q_1	*What are the species, heights, circumferences, and public states of **mature** trees from the **5th arrondissement of Paris**?*	Selection $A\theta c$ Projection
q_2	*What are the species, heights ,circumferences, and stage of development of trees in **woods** around Paris **larger than 250 cm** of circumference? We want the height in **centimeters**.*	Selection $A\theta c$ Projection Math function

query	DB	WSD	Norm.	#c	#c ≥ 2	$\|c\|_{max}$	$\|R^0\|/\|R\|$
R^0				109016	4096	3	1
q_1	0.6 s	6 s	1.3 s	288	1	2	10.2
q_2	0.9 s	9.7 s	1.2 s	75	33	3	1.1

Experiments : Complex selection

q_3	*What are the species, boroughs, heights, ? circumferences, and importance of trees verifying* ***height (cm)*** $<$ ***circumference (cm)*** $\times 5$	Selection $A\theta B$ Projection Math function

query	DB	WSD	Norm.	#c	#c ≥ 2	$\|c\|_{max}$	$\|R^0\|/\|R\|$
R^0				109016	4096	3	1
q_3	0.6 s	339 s	25.2 s	14951	8	2	2.2

Experiments : Complex selection

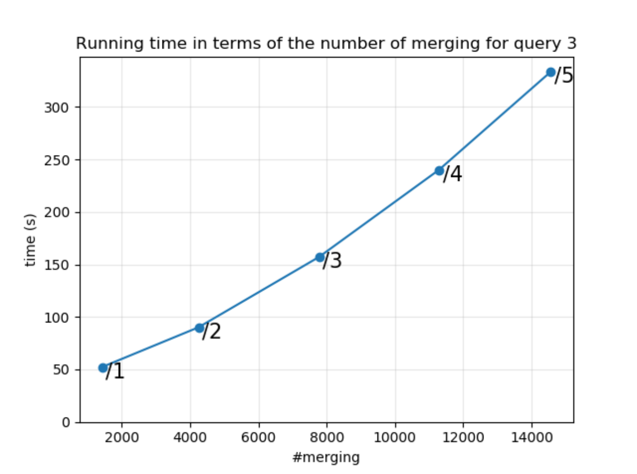


FIGURE – Number of components to merge and running time of the algorithm for different multiplication factors for the circumference in the query

Experiments : Aggregation

q_4	*What is the* ***average height*** *of trees in* q_2 *?*	Aggregation

query	DB	WSD
q_4	0.8 s	3.4

Experiments : Union

q_5	*What are species and circumferences of trees in* $q_1 \cup q_2$ *?*	Union

query	DB	WSD	$\|C\|$	#c	#c ≥ 2	$\|c\|_{max}$	$\|R^0\|/\|R\|$
R^0			465473	109016	4096	3	1
q_5	1.5 s	4.6s	1876	188	0	1	2.8

Experiments : Join

q_6	*Take the **join** of q_1 and q_2 on pairs of trees with different species but the same circumference. What are the species of the two trees and their circumference (+ some other attributes) ?*	Cross product Rename Selection $A\theta B$ Projection

query	DB	WSD	Norm	#c	#c ≥ 2	$\|c\|_{max}$	$\|R^0\|/\|R\|$	
R^0				465473	109016	4096	3	1
q_6	1.7 s	63.8 s	1.5 s	5	3	68	2.4	

$\Rightarrow$ 3 merging steps, but if we change the selection condition in q_2, we get a memory error at the 6^{th} step.

Conclusion

- We saw that we can represent uncertain databases as World-Set Decomposition and how we can implement them in practice, with a finite number of attributes.
- I implemented algorithms described in the paper in `SQL` and I proposed new functions to complete this framework, like aggregation functions.
- I tested my algorithms on a dataset with real uncertainty, the tree dataset.
- We saw that the algorithms output consistent results and need a reasonable amount of time in most cases.

Thanks for your attention !

github.com/TheoDlmz/DataWranglingProject