Counting Problems over Incomplete Databases

Marcelo Arenas, Pablo Barceló, **Mikaël Monet**

June 15th, 2020

Millennium Institute
Foundational
Research on Data

Incomplete databases

- Probabilistic databases: one way of dealing with uncertain data

→ But this is not what is used in practice most of the time...

ProductId	ProductName	Price	Color	Localisation
439	Printer	$100	NULL	Paris center
782	Mouse	$10	red	NULL
398	Mouse	$30	red	Santiago center
...	...	...	...	...

CustomerId	Name	Phone number	Gender	Adress
6	Bob	NULL	male	36 main street
76	Mary	551780726	NULL	NULL
...	...	...	...	...

Incomplete databases

- Probabilistic databases: one way of dealing with uncertain data

→ But this is not what is used in practice most of the time...

ProductId	ProductName	Price	Color	Localisation
439	Printer	$100	NULL	Paris center
782	Mouse	$10	red	NULL
398	Mouse	$30	red	Santiago center
...	...	...	...	...

CustomerId	Name	Phone number	Gender	Adress
6	Bob	NULL	male	36 main street
76	Mary	551780726	NULL	NULL
...	...	...	...	...

→ Incomplete databases: relational databases with **missing values**

How do we query incomplete databases?

- **Default approach** of database theorists for querying incomplete data: certain answers
  - for a *valuation* ν of the nulls of D into constants, let us write $\nu(D)$ the corresponding complete database

  → a tuple $\bar{a}$ is a *certain answer* of $q(\bar{x})$ over the incomplete database D if for every valuation ν of the nulls of D, we have $\bar{a} \in q(\nu(D))$

How do we query incomplete databases?

- **Default approach** of database theorists for querying incomplete data: certain answers
  - for a *valuation* ν of the nulls of D into constants, let us write $\nu(D)$ the corresponding complete database

  → a tuple $\bar{a}$ is a *certain answer* of $q(\bar{x})$ over the incomplete database D if for every valuation ν of the nulls of D, we have $\bar{a} \in q(\nu(D))$

Problem: what if there are no certain answers?

How do we query incomplete databases?

- **Default approach** of database theorists for querying incomplete data: certain answers
  - for a *valuation* ν of the nulls of D into constants, let us write $\nu(D)$ the corresponding complete database

  → a tuple $\bar{a}$ is a *certain answer* of $q(\bar{x})$ over the incomplete database D if for every valuation ν of the nulls of D, we have $\bar{a} \in q(\nu(D))$

Problem: what if there are no certain answers?

→ Recently, Libkin [PODS'18] proposes the notion of better answers

- a tuple $\bar{a}$ is a *better answer* than another tuple $\bar{b}$ if $\{\nu \mid \bar{b} \in q(D)\} \subseteq \{\nu \mid \bar{a} \in q(D)\}$
  → **we can compare (some) tuples**

Another approach: counting

- a tuple $\bar{a}$ is a *better answer* than another tuple $\bar{b}$ if $\{\nu \mid \bar{b} \in q(D)\} \subseteq \{\nu \mid \bar{a} \in q(D)\}$
  - → **we can compare (some) tuples**

To compare all the tuples, why not study the associated counting problems?

Another approach: counting

- a tuple $\bar{a}$ is a *better answer* than another tuple $\bar{b}$ if $\{\nu \mid \bar{b} \in q(D)\} \subseteq \{\nu \mid \bar{a} \in q(D)\}$
  - → **we can compare (some) tuples**

To compare all the tuples, why not study the associated counting problems?

- → "How many valuations ν are such that $\bar{a} \in q(\nu(D))$?"
- → "How many distinct databases of the form $\nu(D)$ are such that $\bar{a} \in q(\nu(D))$?"
  - → **we can compare *all* tuples**

Another approach: counting

- a tuple $\bar{a}$ is a *better answer* than another tuple $\bar{b}$ if $\{\nu \mid \bar{b} \in q(D)\} \subseteq \{\nu \mid \bar{a} \in q(D)\}$
  - → **we can compare (some) tuples**

To compare all the tuples, why not study the associated counting problems?

→ "How many valuations ν are such that $\bar{a} \in q(\nu(D))$?"

→ "How many distinct databases of the form $\nu(D)$ are such that $\bar{a} \in q(\nu(D))$?"

- → **we can compare *all* tuples**

→ **This is what we do!**

Setting

- Incomplete databases with named (marked) nulls
- Each null $\perp$ comes with its own finite domain $\mathrm{dom}(\perp)$; all valuations ν are such that $\nu(\perp) \in \mathrm{dom}(\perp)$
- $\nu(D)$: the (complete) database obtained from D by substituting every null $\perp$ by $\nu(\perp)$, *and then removing duplicate tuples*. We call such a database a completion of D

Setting

- Incomplete databases with named (marked) nulls
- Each null $\perp$ comes with its own finite domain $\mathrm{dom}(\perp)$; all valuations ν are such that $\nu(\perp) \in \mathrm{dom}(\perp)$
- $\nu(D)$: the (complete) database obtained from D by substituting every null $\perp$ by $\nu(\perp)$, *and then removing duplicate tuples*. We call such a database a completion of D

$D =$

R	
$\perp_1$	$\perp_1$
a	$\perp_2$

$\mathrm{dom}(\perp_1) = \{a, b\}$, $\mathrm{dom}(\perp_2) = \{b, c\}$

Setting

- Incomplete databases with named (marked) nulls
- Each null $\perp$ comes with its own finite domain $\mathrm{dom}(\perp)$; all valuations ν are such that $\nu(\perp) \in \mathrm{dom}(\perp)$
- $\nu(D)$: the (complete) database obtained from D by substituting every null $\perp$ by $\nu(\perp)$, *and then removing duplicate tuples*. We call such a database a completion of D

$$D = \begin{array}{|cc|}\hline \multicolumn{2}{|c|}{\mathbf{R}} \\ \hline \perp_1 & \perp_1 \\ a & \perp_2 \\ \hline \end{array} \quad \mathrm{dom}(\perp_1) = \{a, b\},\ \mathrm{dom}(\perp_2) = \{b, c\}$$

$$\nu = \{\perp_1 \mapsto b, \perp_2 \mapsto c\} \quad \rightarrow \quad \nu(D) = \{R(b, b), R(a, c)\}$$

Setting

- Incomplete databases with named (marked) nulls
- Each null $\perp$ comes with its own finite domain $\mathrm{dom}(\perp)$; all valuations ν are such that $\nu(\perp) \in \mathrm{dom}(\perp)$
- $\nu(D)$: the (complete) database obtained from D by substituting every null $\perp$ by $\nu(\perp)$, *and then removing duplicate tuples*. We call such a database a completion of D

$$D = \begin{array}{|cc|}\hline \multicolumn{2}{|c|}{\mathbf{R}} \\ \hline \perp_1 & \perp_1 \\ a & \perp_2 \\ \hline \end{array} \quad \mathrm{dom}(\perp_1) = \{a, b\},\ \mathrm{dom}(\perp_2) = \{b, c\}$$

$$\nu = \{\perp_1 \mapsto b, \perp_2 \mapsto c\} \quad \rightarrow \quad \nu(D) = \{R(b, b), R(a, c)\}$$

$$\nu = \{\perp_1 \mapsto a, \perp_2 \mapsto a\} \quad \rightarrow \quad \nu(D) = \{R(a, a)\}$$

Problems studied

- Fix a Boolean query q

Definition: problem $\#\mathrm{Val}(q)$

Input: an incomplete database D, together with *finite* domains $\mathrm{dom}(\bot)$ for each null of D

Output: the number of valuations ν such that $\nu(D) \models q$

Problems studied

- Fix a Boolean query q

Definition: problem $\#\mathrm{Val}(q)$

Input: an incomplete database D, together with *finite* domains $\mathrm{dom}(\bot)$ for each null of D

Output: the number of valuations ν such that $\nu(D) \models q$

Definition: problem $\#\mathrm{Comp}(q)$

Input: an incomplete database D, together with *finite* domains $\mathrm{dom}(\bot)$ for each null of D

Output: the number of *completions* $\nu(D)$ such that $\nu(D) \models q$

Example

- **Example:** $q = \exists x\, S(x, x)$, $D = \{S(a, b), S(\perp_1, a), S(a, \perp_2)\}$, $\mathrm{dom}(\perp_1) = \{a, b, c\}, \mathrm{dom}(\perp_2) = \{a, b\}$

Example

- **Example:** $q = \exists x\, S(x, x)$, $D = \{S(a, b), S(\bot_1, a), S(a, \bot_2)\}$, $\mathrm{dom}(\bot_1) = \{a, b, c\}, \mathrm{dom}(\bot_2) = \{a, b\}$

$(\nu(\bot_1), \nu(\bot_2))$	(a, a)	(a, b)	(b, a)	(b, b)	(c, a)	(c, b)
$\nu(D)$	S a b a a	S a b a a	S a b b a a a	S a b b a	S a b c a a a	S a b c a
$\nu(D) \models Q$?	Yes	Yes	Yes	No	Yes	No

Example

- **Example:** $q = \exists x\, S(x, x)$, $D = \{S(a, b), S(\bot_1, a), S(a, \bot_2)\}$, $\mathrm{dom}(\bot_1) = \{a, b, c\}, \mathrm{dom}(\bot_2) = \{a, b\}$

$(\nu(\bot_1), \nu(\bot_2))$	(a, a)	(a, b)	(b, a)	(b, b)	(c, a)	(c, b)
$\nu(D)$	S a b a a	S a b a a	S a b b a a a	S a b b a	S a b c a a a	S a b c a
$\nu(D) \models Q$?	Yes	Yes	Yes	No	Yes	No

4 satisfying valuations, 3 satisfying completions

Example

- **Example:** $q = \exists x\, S(x, x)$, $D = \{S(a, b), S(\bot_1, a), S(a, \bot_2)\}$, $\mathrm{dom}(\bot_1) = \{a, b, c\}, \mathrm{dom}(\bot_2) = \{a, b\}$

$(\nu(\bot_1), \nu(\bot_2))$	(a, a)	(a, b)	(b, a)	(b, b)	(c, a)	(c, b)
$\nu(D)$	S $a\ b$ $a\ a$	S $a\ b$ $a\ a$	S $a\ b$ $b\ a$ $a\ a$	S $a\ b$ $b\ a$	S $a\ b$ $c\ a$ $a\ a$	S $a\ b$ $c\ a$
$\nu(D) \models Q$?	Yes	Yes	Yes	No	Yes	No

4 satisfying valuations, 3 satisfying completions

→ Study the complexity of these problems depending on q (*data complexity*). Obtain dichotomies? Can we efficiently approximate the number of solutions? Etc.

Problems variants and query language

- We also study the setting where all labeled nulls are distinct (*Codd tables*; by contrast to *naïve tables*)

- We also study the setting where all nulls share the same domain (*uniform setting*)

→ In total we consider 8 **different problems**

Problems variants and query language

- We also study the setting where all labeled nulls are distinct (*Codd tables*; by contrast to *naïve tables*)

- We also study the setting where all nulls share the same domain (*uniform setting*)

→ In total we consider 8 **different problems**

- We focus on self-join free Boolean conjunctive queries (sjfBCQs)

Results (very simplified)

1. For 7/8 of the variants of our problems, we show a **dichotomy** for $\mathrm{sjfBCQs}$ between $\#\mathrm{P}$-**hard** and in PTIME

2. We show that counting valuations for **Unions of Boolean Conjunctives Queries** always has a fully polynomial-time randomized approximation scheme (FPRAS)

3. We show that counting *completions* does not have a FPRAS

4. We show that counting completions can be SpanP-complete, while it is $\#\mathrm{P}$-complete for counting valuations
   - (SpanP = number of distinct outputs of a nondeterministic Turing machine with output tape running in polynomial time)

Results (very simplified)

1. For 7/8 of the variants of our problems, we show a **dichotomy** for $\mathrm{sjfBCQs}$ between $\#\mathrm{P}$-**hard** and in PTIME

2. We show that counting valuations for **Unions of Boolean Conjunctives Queries** always has a fully polynomial-time randomized approximation scheme (FPRAS)

3. We show that counting *completions* does not have a FPRAS

4. We show that counting completions can be SpanP-complete, while it is $\#\mathrm{P}$-complete for counting valuations
   - (SpanP = number of distinct outputs of a nondeterministic Turing machine with output tape running in polynomial time)

Proof structure (1/2)

Definition: pattern

A sjfBCQ q' is a pattern of another sjfBCQ q if q' can be obtained from q by deleting atoms or variable occurrences, and then reordering the variables inside the atoms and renaming (injectively) the variables and relation names

Proof structure (1/2)

Definition: pattern

A sjfBCQ q' is a pattern of another sjfBCQ q if q' can be obtained from q by deleting atoms or variable occurrences, and then reordering the variables inside the atoms and renaming (injectively) the variables and relation names

Example: $q' = \exists u\, \exists y\, \exists z : R'(u, u, y) \wedge S(z)$ is a pattern of $q = \exists u\, \exists x\, \exists y\, \exists s : R(u, x, u) \wedge S(y, y) \wedge T(x, s, z, s)$

Proof structure (1/2)

Definition: pattern

A sjfBCQ q' is a pattern of another sjfBCQ q if q' can be obtained from q by deleting atoms or variable occurrences, and then reordering the variables inside the atoms and renaming (injectively) the variables and relation names

Example: $q' = \exists u\, \exists y\, \exists z : R'(u, u, y) \wedge S(z)$ is a pattern of $q = \exists u\, \exists x\, \exists y\, \exists s : R(u, x, u) \wedge S(y, y) \wedge T(x, s, z, s)$

$\rightarrow\ R(u, x, u) \wedge S(y, y)$ (delete third atom)

Proof structure (1/2)

Definition: pattern

A sjfBCQ q' is a pattern of another sjfBCQ q if q' can be obtained from q by deleting atoms or variable occurrences, and then reordering the variables inside the atoms and renaming (injectively) the variables and relation names

Example: $q' = \exists u\, \exists y\, \exists z : R'(u, u, y) \wedge S(z)$ is a pattern of $q = \exists u\, \exists x\, \exists y\, \exists s : R(u, x, u) \wedge S(y, y) \wedge T(x, s, z, s)$

- $\rightarrow$ $R(u, x, u) \wedge S(y, y)$ (delete third atom)
- $\rightarrow$ $R(u, x, u) \wedge S(y)$ (delete a variable occurrence)

Proof structure (1/2)

Definition: pattern

A sjfBCQ q' is a pattern of another sjfBCQ q if q' can be obtained from q by deleting atoms or variable occurrences, and then reordering the variables inside the atoms and renaming (injectively) the variables and relation names

Example: $q' = \exists u\, \exists y\, \exists z : R'(u, u, y) \wedge S(z)$ is a pattern of $q = \exists u\, \exists x\, \exists y\, \exists s : R(u, x, u) \wedge S(y, y) \wedge T(x, s, z, s)$

- $\rightarrow$ $R(u, x, u) \wedge S(y, y)$ (delete third atom)
- $\rightarrow$ $R(u, x, u) \wedge S(y)$ (delete a variable occurrence)
- $\rightarrow$ $R(u, u, x) \wedge S(y)$ (reorder variables occurrences)

Proof structure (1/2)

Definition: pattern

A sjfBCQ q' is a pattern of another sjfBCQ q if q' can be obtained from q by deleting atoms or variable occurrences, and then reordering the variables inside the atoms and renaming (injectively) the variables and relation names

Example: $q' = \exists u\, \exists y\, \exists z : R'(u, u, y) \wedge S(z)$ is a pattern of $q = \exists u\, \exists x\, \exists y\, \exists s : R(u, x, u) \wedge S(y, y) \wedge T(x, s, z, s)$

- $\rightarrow$ $R(u, x, u) \wedge S(y, y)$ (delete third atom)
- $\rightarrow$ $R(u, x, u) \wedge S(y)$ (delete a variable occurrence)
- $\rightarrow$ $R(u, u, x) \wedge S(y)$ (reorder variables occurrences)
- $\rightarrow$ $R'(u, u, x) \wedge S(y)$ (rename R into R')

Proof structure (1/2)

Definition: pattern

A sjfBCQ q' is a pattern of another sjfBCQ q if q' can be obtained from q by deleting atoms or variable occurrences, and then reordering the variables inside the atoms and renaming (injectively) the variables and relation names

Example: $q' = \exists u\, \exists y\, \exists z : R'(u, u, y) \wedge S(z)$ is a pattern of $q = \exists u\, \exists x\, \exists y\, \exists s : R(u, x, u) \wedge S(y, y) \wedge T(x, s, z, s)$

- $\rightarrow R(u, x, u) \wedge S(y, y)$ (delete third atom)
- $\rightarrow R(u, x, u) \wedge S(y)$ (delete a variable occurrence)
- $\rightarrow R(u, u, x) \wedge S(y)$ (reorder variables occurrences)
- $\rightarrow R'(u, u, x) \wedge S(y)$ (rename R into R')
- $\rightarrow R'(u, u, y) \wedge S(z)$ (rename x into y and y into z)

Proof structure (2/2)

Lemma

Let q, q' be $\mathrm{sjfBCQs}$ such that q' is a pattern of q. Then we have $\#\mathsf{Val}(q') \leq^{\mathrm{P}} \#\mathsf{Val}(q)$

- (and the same results holds for counting completions, and also if we restrict to Codd tables and/or to the uniform setting)

Proof structure (2/2)

Lemma

Let q, q' be $\mathrm{sjfBCQs}$ such that q' is a pattern of q. Then we have $\#\mathrm{Val}(q') \leq^{\mathrm{P}} \#\mathrm{Val}(q)$

- (and the same results holds for counting completions, and also if we restrict to Codd tables and/or to the uniform setting)

→ for each variant of the problem, find a set of patterns that are hard and such that if a sjfBCQ does not have any of these patterns then the problem is in PTIME

The hard patterns

	Counting valuations		Counting completions	
	Non-uniform	**Uniform**	**Non-uniform**	**Uniform**
Naïve	$R(x,x)$ $R(x) \wedge S(x)$	$R(x,x)$ $R(x) \wedge S(x,y) \wedge T(y)$ $R(x,y) \wedge S(x,y)$	$R(x)$	$R(x,x)$ $R(x,y)$
Codd	$R(x) \wedge S(x)$	$R(x) \wedge S(x,y) \wedge T(y)$?	$R(x)$	$R(x,x)$ $R(x,y)$

The hard patterns

	Counting valuations		Counting completions	
	Non-uniform	Uniform	Non-uniform	Uniform
Naïve	$R(x,x)$ $R(x) \wedge S(x)$	$R(x,x)$ $R(x) \wedge S(x,y) \wedge T(y)$ $R(x,y) \wedge S(x,y)$	$R(x)$	$R(x,x)$ $R(x,y)$
Codd	$R(x) \wedge S(x)$	$R(x) \wedge S(x,y) \wedge T(y)$?	$R(x)$	$R(x,x)$ $R(x,y)$

→ Valuations, non-uniform, naïve: each variable has only one occurrence (example: for $\#\mathrm{Val}(\exists x : R(x,x))$, reduction from #3-coloring)

The hard patterns

	Counting valuations		Counting completions	
	Non-uniform	**Uniform**	**Non-uniform**	**Uniform**
Naïve	$R(x,x)$ $R(x) \wedge S(x)$	$R(x,x)$ $R(x) \wedge S(x,y) \wedge T(y)$ $R(x,y) \wedge S(x,y)$	$R(x)$	$R(x,x)$ $R(x,y)$
Codd	$R(x) \wedge S(x)$	$R(x) \wedge S(x,y) \wedge T(y)$?	$R(x)$	$R(x,x)$ $R(x,y)$

→ Valuations, non-uniform, naïve: each variable has only one occurrence (example: for $\#\mathrm{Val}(\exists x : R(x,x))$, reduction from #3-coloring)

→ Valuations, non-uniform, Codd: each variable occurs in at most one atom

The hard patterns

	Counting valuations		Counting completions	
	Non-uniform	**Uniform**	**Non-uniform**	**Uniform**
Naïve	$R(x,x)$ $R(x) \wedge S(x)$	$R(x,x)$ $R(x) \wedge S(x,y) \wedge T(y)$ $R(x,y) \wedge S(x,y)$	$R(x)$	$R(x,x)$ $R(x,y)$
Codd	$R(x) \wedge S(x)$	$R(x) \wedge S(x,y) \wedge T(y)$?	$R(x)$	$R(x,x)$ $R(x,y)$

→ Valuations, non-uniform, naïve: each variable has only one occurrence (example: for $\#\mathrm{Val}(\exists x : R(x,x))$, reduction from #3-coloring)

→ Valuations, non-uniform, Codd: each variable occurs in at most one atom

→ Completions, uniform (naïve or Codd): all the atoms are unary

The hard patterns

	Counting valuations		Counting completions	
	Non-uniform	Uniform	Non-uniform	Uniform
Naïve	$R(x,x)$ $R(x) \wedge S(x)$	$R(x,x)$ $R(x) \wedge S(x,y) \wedge T(y)$ $R(x,y) \wedge S(x,y)$	$R(x)$	$R(x,x)$ $R(x,y)$
Codd	$R(x) \wedge S(x)$	$R(x) \wedge S(x,y) \wedge T(y)$?	$R(x)$	$R(x,x)$ $R(x,y)$

→ Valuations, non-uniform, naïve: each variable has only one occurrence (example: for $\#\mathrm{Val}(\exists x : R(x,x))$, reduction from #3-coloring)

→ Valuations, non-uniform, Codd: each variable occurs in at most one atom

→ Completions, uniform (naïve or Codd): all the atoms are unary

(So. . . not much is tractable)

Conclusion

To sum up:

- Counting valuations and completions is **hard**, even in very restricted settings (uniform Codd tables)
- But counting valuations has a **FPRAS** for UCQs
- (while counting completions does not)
- SpanP is the right class to consider for problems of the form #Comp(q)

Conclusion

To sum up:

- Counting valuations and completions is **hard**, even in very restricted settings (uniform Codd tables)
- But counting valuations has a **FPRAS** for UCQs
- (while counting completions does not)
- SpanP is the right class to consider for problems of the form #Comp(q)

Future work:

- Complete our 8th dichotomy
- Extend the results to CQs? UCQs?
- Use knowledge compilation to capture the tractability of the tractable queries? (as is done for probabilistic databases)

Conclusion

To sum up:

- Counting valuations and completions is **hard**, even in very restricted settings (uniform Codd tables)
- But counting valuations has a **FPRAS** for UCQs
- (while counting completions does not)
- SpanP is the right class to consider for problems of the form #Comp(q)

Future work:

- Complete our 8th dichotomy
- Extend the results to CQs? UCQs?
- Use knowledge compilation to capture the tractability of the tractable queries? (as is done for probabilistic databases)

Thanks for your attention!

Bibliography I


Leonid Libkin.
Certain Answers Meet Zero-One Laws.
In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 195–207, 2018.