Saturday 31st March, 2018

minimum-QID

INSTANCE
A finite set of m-tuples of database $\mathcal{D}$ with an m-tuple of column names $\mathcal{C}$ and an m-tuple of finite sets of column domains $\mathcal{F}$, such that $\mathcal{D} \subseteq \bigtimes_{F \in \mathcal{F}} F$ and, for all $F \in \mathcal{F}$, $|\mathcal{D}| \leqslant |F|$.

QUESTION
What is the finite set $I \subseteq \mathrm{set}(\mathcal{C})$ with minimal $|I|$ such that $|\Pi_I D| = |D|$?

Cf. **k-QID**:

QUESTION
Is there a finite set $I \subseteq \mathrm{set}(\mathcal{C})$ with $|I| = k$ such that $|\Pi_I D| = |D|$?

Algorithm

```
Data: A database.
Result: A set of columns that completely identifies the database.
$goal = |\Pi_{\mathcal{C}} \mathcal{D}|$
$fields = \mathcal{C}$
$found = True$
while $found$ do
    $found = False$
    for $next \in combinations(fields, |fields| - 1)$ do
        if $|\Pi_{next} \mathcal{D}| = goal$ then
            $found = True$
            $fields = next$
            break
return $fields$
```

The running time is $\mathcal{O}(\Sigma_{i=1}^{|\mathcal{D}|} \binom{i}{i-1}) = \mathcal{O}(|\mathcal{D}|^2)$.

Example

Consider

a	b	c	d	e
x	1	y	y	z
x	x	y	2	z
x	3	y	y	z
x	x	y	4	z
x	5	y	y	z

The correct minimum QID is (b,d) and the algorithm finds it by choosing

a	b	c	d	e	yes
a	b	c	d		yes
a	b	c			no
	b	c	d		yes
	b	c			no
	b		d		yes
	b				no
			d		no

It's wrong

The function problem is at least as hard as decision problem. So it must be that the algorithm is either incorrect or superpolynomial. It turns out that it is incorrect. It does not solve the **minimum-QID**. Instead, it solves something else.

It finds a "minimal QID" C such that for all $C' \subsetneq C$, C' is not a QID. So it may not find the global optimum but a local one.

Counter-example

Consider

a	b	c	d	e
x	1	x	6	x
x	x	2	7	x
x	3	x	8	x
x	x	4	x	9
5	x	x	x	0

The correct minimum QID is (d,e) and the algorithm finds (a,b,c)

a	b	c	d	e	yes
a	b	c	d		yes
a	b	c			yes
a	b				no
	b	c			no
a		c			no

So, what now?

This alone doesn't really look to me like something that can be turned into a randomized or approximation algorithm with useful bounds, but I don't think this is useless. Also, is this some sort of heuristics? For example, what if we allow backtracking? Such as

⋮

a		c	d		yes
a		c			no
		c	d		no
a			d		no
a	b		d		no
	b	c	d		no
	b	c	d	e	yes

⋮