Certain Conjunctive Query Answering in SQL

Alexandre Decan, Fabian Pijcke, and Jef Wijsen

Université de Mons, Mons, Belgium,
{alexandre.decan, jef.wijsen}@umons.ac.be, fabian.pijcke@gmail.com

Abstract. An uncertain database **db** is defined as a database in which distinct tuples of the same relation can agree on their primary key. A repair (or possible world) of **db** is then obtained by selecting a maximal number of tuples without ever selecting two distinct tuples of the same relation that agree on their primary key. Given a query Q on **db**, the *certain answer* is the intersection of the answers to Q on all repairs. Recently, a syntactic characterization was obtained of the class of acyclic self-join-free conjunctive queries for which certain answers are definable by a first-order formula, called certain first-order rewriting [15]. In this article, we investigate the nesting and alternation of quantifiers in certain first-order rewritings, and propose two syntactic simplification techniques. We then experimentally verify whether these syntactic simplifications result in lower execution times on real-life SQL databases.

1 Introduction

Uncertainty can be modeled in the relational model by allowing primary key violations. Primary keys are underlined in the conference planning database $\mathbf{db}_0$ in Fig. 1. There are still two candidate cities for organizing SUM 2016 and SUM 2017. The table S shows controversy about the attractiveness of Mons, while information about the attractiveness of Gent is missing. A *repair* (or possible world) is obtained by selecting a maximal number of tuples, without selecting two tuples with the same primary key value. Database $\mathbf{db}_0$ has 8 repairs, because there are two choices for SUM 2016, two choices for SUM 2017, and two

R	Conf	Year	Town
	SUM	2012	Marburg
	SUM	2016	Mons
	SUM	2016	Gent
	SUM	2017	Rome
	SUM	2017	Paris

S	Town	Attractiveness
	Charleroi	C
	Marburg	A
	Mons	A
	Mons	B
	Paris	A
	Rome	A

Fig. 1. Uncertain database $\mathbf{db}_0$.

choices for Mons' attractiveness. The following conjunctive query asks in which years SUM took place (or will take place) in a city with A attractiveness:

$$Q_0 = \{y \mid \exists z \big(\mathsf{R}(\underline{\text{`SUM'}, y}, z) \land \mathsf{S}(\underline{z}, \text{`A'})\big)\}.$$

The *certain (query) answer* is the intersection of the query answers on all repairs, which in this example is $\{2012, 2017\}$. Notice incidentally that $Q_0(\mathbf{db}_0)$ also contains 2016, but that answer is not certain, because in some repairs, the organizing city of SUM 2016 does not have A attractiveness.

For every database, the certain answer to Q_0 is obtained by the following first-order query:

$$\varphi_0 = \{y \mid \exists z \mathsf{R}(\underline{\text{`SUM'}, y}, z) \land \forall z \left( \mathsf{R}(\underline{\text{`SUM'}, y}, z) \to \begin{bmatrix} \mathsf{S}(\underline{z}, \text{`A'}) \land \\ \forall v \big(\mathsf{S}(\underline{z}, v) \to v = \text{`A'}\big) \end{bmatrix} \right)\}.$$

We call φ_0 a certain first-order rewriting for Q_0. Certain first-order rewritings are of practical importance, because they can be encoded in SQL, which allows to obtain certain answers using standard database technology. However, it is well-known that not all conjunctive queries have a certain first-order rewriting [4, 16], and it remains an open problem to syntactically characterize the conjunctive queries that have one. Nevertheless, for conjunctive queries that are acyclic and self-join-free, such characterization has recently been found [15, 17].

In this article, we focus on the class of acyclic self-join free conjunctive queries that have a certain first-order rewriting. We first provide algorithm NaiveFo which takes such query as input, and constructs its certain first-order rewriting. We then provide two theorems indicating that rewritings produced by algorithm NaiveFo can generally be "simplified" by (i) reducing the number of (alternations of) quantifier blocks and/or by (ii) reducing the quantifier nesting depth. Finally, the implementation of our theory shows that certain SQL rewriting is an effective and efficient technique for computing certain answers.

This article is organized as follows. Section 2 provides notations and definitions. In particular, we provide measures for describing the syntactic complexity of a first-order formula. Section 3 discusses related work. Section 4 introduces the construct of *attack graph* which is essential for the purpose of certain first-order rewriting. Section 5 gives the code of algorithm NaiveFo. Section 6 shows how rewritings can be simplified with respect to the complexity measures of Section 2. Section 7 reports on our experiments conducted on real-life SQL databases. Section 8 concludes the article.

2 Notations and Terminology

We assume a set of *variables* disjoint from a set **dom** of *constants*. We will assume some fixed total order on the set of variables, which will only serve to "serialize" sets of variables into sequences in a unique way. If $\boldsymbol{x}$ is a sequence of variables and constants, then $\mathsf{vars}(\boldsymbol{x})$ is the set of variables that occur in $\boldsymbol{x}$.

Let U be a set of variables. A *valuation over* U is a total mapping θ from U to **dom**. Such valuation θ is often extended to be the identity on constants and on variables not in U.

Key-equal atoms Every *relation name* R has a fixed *signature*, which is a pair $[n, k]$ with $n \geq k \geq 1$: the integer n is the *arity* of the relation name and $\{1, 2, \ldots, k\}$ is the *primary key*. If R is a relation name with signature $[n, k]$, then $R(s_1, \ldots, s_n)$ is an R*-atom* (or simply atom), where each s_i is a constant or a variable ($1 \leq i \leq n$). Such atom is commonly written as $R(\underline{\boldsymbol{x}}, \boldsymbol{y})$ where the primary key value $\boldsymbol{x} = s_1, \ldots, s_k$ is underlined and $\boldsymbol{y} = s_{k+1}, \ldots, s_n$. An atom is *ground* if it contains no variables. Two ground atoms $R_1(\underline{\boldsymbol{a}_1}, \boldsymbol{b}_1), R_2(\underline{\boldsymbol{a}_2}, \boldsymbol{b}_2)$ are *key-equal* if $R_1 = R_2$ and $\boldsymbol{a}_1 = \boldsymbol{a}_2$. The arity of an atom F, denoted $\mathsf{arity}(F)$, is the arity of its relation name.

Database and repair A *database schema* is a finite set of *relation names*. All constructs that follow are defined relative to a fixed database schema.

A *database* is a finite set $\mathbf{db}$ of ground atoms using only the relation names of the schema. Importantly, a database can contain distinct, key-equal atoms. Intuitively, if a database contains distinct, key-equal atoms A and B, then only one of A or B can be true, but we do not know which one. In this respect, the database contains uncertainty. A database $\mathbf{db}$ is *consistent* if it does not contain two distinct atoms that are key-equal. A *repair* of a database $\mathbf{db}$ is a maximal (under set inclusion) consistent subset of $\mathbf{db}$.

Conjunctive queries A *conjunctive query* is a pair (q, V) where $q = \{R_1(\underline{\boldsymbol{x}_1}, \boldsymbol{y}_1), \ldots, R_n(\underline{\boldsymbol{x}_n}, \boldsymbol{y}_n)\}$ is a finite set of atoms and V is a subset of the variables occurring in q. Every variable of V is *free*; the other variables are *bound*. This query represents the first-order formula $\exists u_1 \ldots \exists u_k \big(R_1(\underline{\boldsymbol{x}_1}, \boldsymbol{y}_1) \wedge \cdots \wedge R_n(\underline{\boldsymbol{x}_n}, \boldsymbol{y}_n)\big)$, in which $u_1, \ldots, u_k$ are all the variables of $\mathsf{vars}(\boldsymbol{x}_1\boldsymbol{y}_1 \ldots \boldsymbol{x}_n\boldsymbol{y}_n) \setminus V$. If $V = \emptyset$, then Q is called *Boolean*. We write $\mathsf{vars}(q)$ for the set of variables that occur in q. We denote by $|q|$ the number of atoms in q, and we define $\mathsf{aritysum}(q) = \sum_{F \in q} \mathsf{arity}(F)$.

Let $\mathbf{db}$ be a database. Let $Q = (q, V)$ be a conjunctive query, and let $U = \mathsf{vars}(q)$. Let $\boldsymbol{x} = \langle x_1, \ldots, x_m \rangle$ be the variables of V ordered according to the total order on the set of variables. The *answer* to Q on $\mathbf{db}$, denoted $Q(\mathbf{db})$, is defined as follows:

$$Q(\mathbf{db}) = \{\theta(\boldsymbol{x}) \mid \theta \text{ is a valuation over } U \text{ such that } \theta(q) \subseteq \mathbf{db}\}.$$

In particular, if Q is Boolean, then either $Q(\mathbf{db}) = \{\langle\rangle\}$ (representing **true**) or $Q(\mathbf{db}) = \{\}$ (representing **false**).

We say that Q has a *self-join* if some relation name occurs more than once in q; if Q has no self-join, then it is called *self-join-free*. We write SJFCQ for the class of self-join-free conjunctive queries.

Certain conjunctive query answering Let $Q = (q, V)$ be a conjunctive query. The *certain answer* to Q on $\mathbf{db}$, denoted $Q_{\mathsf{sure}}(\mathbf{db})$, is defined as follows:

$$Q_{\mathsf{sure}}(\mathbf{db}) = \bigcap\{Q(\mathbf{rep}) \mid \mathbf{rep} \text{ is a repair of } \mathbf{db}\}.$$

Let $\langle x_1, \ldots, x_m \rangle$ be the ordered sequence of variables in V. We say that the certain answer to Q is *first-order computable* if there exists a first-order formula

$\varphi(x_1, \ldots, x_m)$, with free variables x_1, …, x_m, such that for every database $\mathbf{db}$, for every $\boldsymbol{a} \in \mathbf{dom}^m$, $\boldsymbol{a} \in Q_{\mathsf{sure}}(\mathbf{db}) \iff \mathbf{db} \models \varphi(\boldsymbol{a})$. The formula φ, if it exists, is called a *certain first-order rewriting* for Q; its encoding in SQL is a *certain SQL rewriting*.

Notational conventions We use letters F, G, H, I for atoms appearing in a query. For $F = R(\underline{\boldsymbol{x}}, \boldsymbol{y})$, we denote by $\mathsf{KeyVars}(F)$ the set of variables that occur in $\boldsymbol{x}$, and by $\mathsf{Vars}(F)$ the set of variables that occur in F, that is, $\mathsf{KeyVars}(F) = \mathsf{vars}(\boldsymbol{x})$ and $\mathsf{Vars}(F) = \mathsf{vars}(\boldsymbol{x}) \cup \mathsf{vars}(\boldsymbol{y})$.

Acyclic conjunctive queries A *join tree* τ for a conjunctive query $Q = (q, V)$ is an undirected tree whose vertices are the atoms of q such that whenever the same variable x occurs in two atoms F and G, then either $x \in V$ (i.e., x is free) or x occurs in each atom on the unique path linking F and G. We will assume that join trees are edge-labeled, such that an edge between F and G is labeled with the set $\big(\mathsf{Vars}(F) \cap \mathsf{Vars}(G)\big) \setminus V$. A conjunctive query Q is called *acyclic* if it has a join tree [2].

Quantifier rank and quantifier alternation depth The *quantifier rank* of a first-order formula φ, denoted by $\mathsf{qr}(\varphi)$, is the depth of the quantifier nesting in φ and is defined as usual (see, for example, [10, page 32]):

- If φ is quantifier-free, then $\mathsf{qr}(\varphi) = 0$.
- $\mathsf{qr}(\varphi_1 \wedge \varphi_2) = \mathsf{qr}(\varphi_1 \vee \varphi_2) = \max\big(\mathsf{qr}(\varphi_1), \mathsf{qr}(\varphi_2)\big)$;
- $\mathsf{qr}(\neg\varphi) = \mathsf{qr}(\varphi)$;
- $\mathsf{qr}(\exists x \varphi) = \mathsf{qr}(\forall x \varphi) = 1 + \mathsf{qr}(\varphi)$.

A first-order formula φ is said to be in *prenex normal form* if it has the form $Q_1 x_1 \ldots Q_n x_n \psi$, where Q_i's are either $\exists$ or $\forall$ and ψ is quantifier-free. We say that φ has *quantifier alternation depth* m if $Q_1 x_1 \ldots Q_n x_n$ can be divided into m blocks such that all quantifiers in a block are of the same type and quantifiers in two consecutive blocks are different.

For formulas not in prenex normal form, the number of quantifier blocks is counted as follows. A universally quantified formula is a formula whose main connective is $\forall$. An existentially quantified formula is a formula whose main connective is $\exists$. The number of quantifier blocks in a first-order formula φ, denoted $\mathsf{qbn}(\varphi)$, is defined as follows.

- If φ is quantifier-free, then $\mathsf{qbn}(\varphi) = 0$.
- $\mathsf{qbn}(\varphi_1 \wedge \varphi_2) = \mathsf{qbn}(\varphi_1 \vee \varphi_2) = \mathsf{qbn}(\varphi_1) + \mathsf{qbn}(\varphi_2)$;
- $\mathsf{qbn}(\neg\varphi) = \mathsf{qbn}(\varphi)$;
- if φ is not universally quantified and $n \geq 1$, then $\mathsf{qbn}(\forall x_1 \ldots \forall x_n \varphi) = 1 + \mathsf{qbn}(\varphi)$; and
- if φ is not existentially quantified and $n \geq 1$, then $\mathsf{qbn}(\exists x_1 \ldots \exists x_n \varphi) = 1 + \mathsf{qbn}(\varphi)$.

For example, if φ is $\exists x \exists y(\exists u \varphi_1 \wedge \exists v \varphi_2)$ and φ_1, φ_2 are both quantifier-free, then $\mathsf{qbn}(\varphi) = 3$. Notice that φ has a prenex normal form with quantifier alternation depth equal to 1. Clearly, if φ is in prenex normal form, then the quantifier alternation depth of φ is equal to $\mathsf{qbn}(\varphi)$.

Proposition 1. *Every first-order formula φ has an equivalent one in prenex normal form with quantifier alternation depth less than or equal to $\mathsf{qbn}(\varphi)$.*

3 Related Work

Certain (or consistent) query answering was founded in the seminal work by Arenas, Bertossi, and Chomicki [1]. The current state of the art can be found in [3]. Fuxman and Miller [8] were the first ones to focus on certain first-order rewriting of SJFCQ queries under primary key constraints, with applications in the ConQuer system [7]. Their results have been generalized by Wijsen [15, 17], who obtained a syntactic characterization of those acyclic SJFCQ queries that allow certain first-order rewriting. It follows from the proof of Corollary 5 in [14] that acyclicity is also implicit in the work of Fuxman and Miller.

Given a Boolean query Q, $\mathsf{CERTAINTY}(Q)$ is the decision problem that takes as input an (uncertain) database **db** and asks whether Q evaluates to true on every repair. Wijsen [16] showed that the class SJFCQ contains Boolean queries Q such that $\mathsf{CERTAINTY}(Q)$ is in **P** but not first-order expressible. It is an open conjecture that for every Boolean SJFCQ query Q, it is the case that $\mathsf{CERTAINTY}(Q)$ is in **P** or **coNP**-complete. For queries with exactly two atoms, such dichotomy was recently shown true [9].

Maslowski and Wijsen [12, 11] have studied the complexity of the counting variant of $\mathsf{CERTAINTY}(Q)$, denoted $\natural\mathsf{CERTAINTY}(Q)$. Given a database **db**, the problem $\natural\mathsf{CERTAINTY}(Q)$ asks to determine the exact number of repairs of **db** that satisfy some Boolean query Q. They showed that that for every Boolean SJFCQ query Q, it is the case that $\natural\mathsf{CERTAINTY}(Q)$ is in **P** or $\natural$**P**-complete. The problem $\natural\mathsf{CERTAINTY}(Q)$ is closely related to query answering in probabilistic data models [5]. From the probabilistic database angle, our uncertain databases are a restricted case of *block-independent-disjoint* probabilistic databases [5, 6]. A *block* in a database **db** is a maximal subset of key-equal atoms. If $\{R(\underline{\boldsymbol{a}}, \boldsymbol{b}_1), \ldots, R(\underline{\boldsymbol{a}}, \boldsymbol{b}_n)\}$ is a block of size n, then every atom of the block has a probability of $1/n$ to be selected in a repair of **db**. Every repair is a possible world, and all these worlds have the same probability.

4 Attack Graph

Let $Q = (q, V)$ be an acyclic SJFCQ query and $U = \mathsf{vars}(q)$. For every $F \in q$, we denote by $\mathsf{FD}(Q, F)$ the set of functional dependencies that contains $X \to Y$ whenever q contains some atom G with $G \neq F$ such that $X = \mathsf{KeyVars}(G) \setminus V$ and $Y = \mathsf{Vars}(G) \setminus V$. For every $F \in q$, we define:

$$F^{+,Q} = \{x \in U \mid \mathsf{FD}(Q, F) \models \big(\mathsf{KeyVars}(F) \setminus V\big) \to x\}.$$

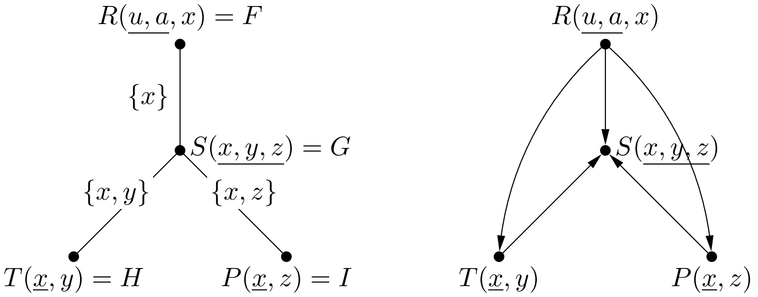

Fig. 2. Join tree (left) and attack graph (right) for Boolean query $Q = (q, \emptyset)$ with $q = \{R(\underline{u,a},x), S(\underline{x,y},z), T(\underline{x},y), P(\underline{x},z)\}$. It is understood that a is a constant.

Example 1. For the Boolean query Q in Fig. 2, we have $\mathsf{FD}(Q,F) \equiv \{x \rightarrow y, x \rightarrow z\}$, $\mathsf{FD}(Q,G) \equiv \{u \rightarrow x, x \rightarrow y, x \rightarrow z\}$, $\mathsf{FD}(Q,H) \equiv \{u \rightarrow x, x \rightarrow z\}$, and $\mathsf{FD}(Q,I) \equiv \{u \rightarrow x, x \rightarrow y\}$. It follows $F^{+,Q} = \{u\}$, $G^{+,Q} = \{x,y,z\}$, $H^{+,Q} = \{x,z\}$, and $I^{+,Q} = \{x,y\}$. More elaborated examples can be found in [15].

Let τ be a join tree for Q. The *attack graph* of τ is a directed graph whose vertices are the atoms of q. There is a directed edge from F to some atom G if for every edge e on the unique path in τ that links F and G, there exists a variable x in e's edge label such that $x \notin F^{+,Q}$.

Example 2. See Fig. 2 (right). There is a directed edge from F to H because the edge labels $\{x\}$ and $\{x,y\}$ on the path between F and H are not contained in $F^{+,Q} = \{u\}$. The attack graph is acyclic.

It is known [17] that if τ_1 and τ_2 are two join trees for Q, then the attack graphs of τ_1 and τ_2 are identical. The attack graph of Q is then defined as the attack graph of any join tree for Q. If the attack graph contains a directed edge from atom F to atom G, then we say that F *attacks* G. An atom F of Q is said to be *unattacked* if the attack graph of Q contains no directed edge that points to F (i.e., F has zero indegree).

The following result was shown in [15] for Boolean acyclic SJFCQ queries. The extension to queries with free variables is easy and already indicated in [15].

Theorem 1. *Let $Q = (q, V)$ be an acyclic SJFCQ query. Then, Q has a certain first-order rewriting if and only if the attack graph of Q is acyclic.*

5 Naive Algorithm

Algorithm NaiveFo implements definitions given in [15]. The algorithm takes two inputs: an acyclic SJFCQ query $Q = (q, V)$ and a directed acyclic graph

Function NaiveFo(q,V,E) Construct certain first-order rewriting

```
Input: $Q = (q, V)$ is an acyclic $\mathsf{SJFCQ}$ query, where $V$ is the set of free variables of $q$.
       $E \subseteq q \times q$ is an acyclic set of directed edges containing the attack graph of $Q$.
Result: certain first-order rewriting $\varphi$ for $Q$.
begin
    if $q = \emptyset$ then
        $\varphi \leftarrow$ true;
    else
        choose an atom $F = R(\underline{x_1, \ldots, x_k}, y_1, \ldots, y_\ell)$ that is unattacked in $E$;
        $V' \leftarrow V$;
        $\mathsf{X} \leftarrow \emptyset$;
        foreach $i \leftarrow 1$ to $k$ do
            if $x_i$ is a variable and $x_i \notin V'$ then
                $V' \leftarrow V' \cup \{x_i\}$;
                $\mathsf{X} \leftarrow \mathsf{X} \cup \{x_i\}$;
        $\mathsf{Y} \leftarrow \emptyset$;
        $\mathsf{NEW} \leftarrow \emptyset$;
        foreach $i \leftarrow 1$ to $\ell$ do
            if $y_i$ is a constant or $y_i \in V'$ then
                let $z_i$ be a new variable;
                $\mathsf{NEW} \leftarrow \mathsf{NEW} \cup \{i\}$;
            else                                         /* $y_i$ is a variable not in $V'$ */
                let $z_i$ be the same variable as $y_i$;
                $V' \leftarrow V' \cup \{y_i\}$;
                $\mathsf{Y} \leftarrow \mathsf{Y} \cup \{y_i\}$;
        $q' \leftarrow q \setminus \{F\}$;
        $E' \leftarrow E \setminus (\{F\} \times q)$;
        $V' \leftarrow V' \cap \mathsf{vars}(q')$;
        $\varphi \leftarrow \exists \mathsf{X} \left[ \begin{array}{l} \exists \mathsf{Y} R(\underline{x_1, \ldots, x_k}, y_1, \ldots, y_\ell) \wedge \\ \forall z_1 \ldots \forall z_\ell \Big( R(\underline{x_1, \ldots, x_k}, z_1, \ldots, z_\ell) \rightarrow \left[ \begin{array}{l} \bigwedge_{i \in \mathsf{NEW}} z_i = y_i \wedge \\ NaiveFo(q', V', E') \end{array} \right] \Big) \end{array} \right]$;
    return $\varphi$;
```

that contains Q's attack graph. The algorithm computes a certain first-order rewriting φ for Q. For example, the formula φ_0 in Section 1 was obtained by applying the algorithm on Q_0.

Algorithm NaiveFo is called "naive" because it does not attempt to minimize the alternations or nesting depth of quantifiers. This may be problematic when the rewritings are translated into SQL for execution, as illustrated by the following example.

Example 3. For each $m \geq 1$, assume relation name R_i with signature $[2,1]$, and let $\lfloor m \rfloor = \{R_1(\underline{x_1}, b), \ldots, R_m(\underline{x_m}, b)\}$, where b is a constant. For $m \geq 1$, let $\lfloor\!\lfloor m \rfloor\!\rfloor = (\lfloor m \rfloor, \emptyset)$, a Boolean query whose attack graph has no edges. Formulas φ_1, φ_2, and φ_3 are three possible certain first-order rewritings for $\lfloor\!\lfloor m \rfloor\!\rfloor$. The formula φ_1 is returned by algorithm NaiveFo, while φ_2 and φ_3 result from some syntactic simplification techniques described in Section 6. In particular, φ_2 minimizes the number of quantifier blocks, and φ_3 minimizes the nesting depth of quantifiers.

$$\begin{aligned}
\varphi_1 = \exists x_1 \Bigg(& R_1(\underline{x_1}, b) \wedge \forall z_1 \Bigg(R_1(\underline{x_1}, z_1) \rightarrow z_1 = b \wedge \\
& \exists x_2 \Bigg(R_2(\underline{x_2}, b) \wedge \forall z_2 \Bigg(R_2(\underline{x_2}, z_2) \rightarrow z_2 = b \wedge \\
& \quad \ddots \\
& \qquad \exists x_m \big(R_m(\underline{x_m}, b) \wedge \forall z_m \big(R_m(\underline{x_m}, z_m) \rightarrow z_m = b \big) \big) \ldots \Bigg) \Bigg) \Bigg) \Bigg) \\
\varphi_2 = {} & \exists x_1 \ldots \exists x_m \Big(\bigwedge_{i=1}^{m} R_i(\underline{x_i}, b) \wedge \forall z_1 \ldots \forall z_m \big(\bigwedge_{i=1}^{m} R_i(\underline{x_i}, z_i) \rightarrow \bigwedge_{i=1}^{m} z_i = b \big) \Big) \\
\varphi_3 = {} & \bigwedge_{i=1}^{m} \exists x_i \Big(R_i(\underline{x_i}, b) \wedge \forall z_i \big(R_i(\underline{x_i}, z_i) \rightarrow z_i = b \big) \Big)
\end{aligned}$$

Notice that φ_1, φ_2, and φ_3 each contain m existential and m universal quantifiers. The following table gives the quantifier rank and the number of quantifier blocks for these formulas; recall that these measures were defined in Section 2.

i	$\mathsf{qr}(\varphi_i)$	$\mathsf{qbn}(\varphi_i)$
1	$2m$	$2m$
2	$2m$	2
3	2	$2m$

The differences in syntactic complexity persist in SQL. Assume that for each $i \in \{1, \ldots, m\}$, the first and the second attribute of each R_i are named A and B respectively. Thus, A is the primary key attribute. For $m = 2$, the queries `Q1`, `Q2`, and `Q3` in Fig. 3 are direct translations into SQL of φ_1, φ_2, and φ_3.[1] The

[1] In practice, we construct rewritings in tuple relational calculus (TRC), and then translate TRC into SQL. Such translations are well known (see, e.g., Chapter 3 of [13]). We omit the details of these translations in this article because of space limitations.

```
Q1 = SELECT ’true’ FROM R1 AS r11
     WHERE  NOT EXISTS (SELECT * FROM R1 AS r12
                        WHERE  r12.A = r11.A
                        AND   (r12.B <> ’b’
                        OR     NOT EXISTS (SELECT * FROM R2 AS r21
                                           WHERE  NOT EXISTS (SELECT * FROM R2 AS r22
                                                              WHERE  r22.A = r21.A
                                                              AND    r22.B <> ’b’))))

Q2 = SELECT ’true’ FROM R1 AS r11, R2 AS r21
     WHERE  NOT EXISTS (SELECT * FROM R1 AS r12, R2 AS r22
                        WHERE  r12.A = r11.A AND r22.A = r21.A
                        AND   (r12.B <> ’b’ OR r22.B <> ’b’))

Q3 = SELECT ’true’ FROM R1 AS r11
     WHERE  NOT EXISTS (SELECT * FROM R1 AS r12
                        WHERE  r12.A = r11.A AND r12.B <> ’b’)
     INTERSECT
     SELECT ’true’ FROM R2 AS r21
     WHERE  NOT EXISTS (SELECT * FROM R2 AS r22
                        WHERE  r22.A = r21.A AND r22.B <> ’b’)
```

Fig. 3. Q1, Q2, Q3 are certain SQL rewritings for $[\![2]\!]$.

fact that φ_2 only has one $\forall$ quantifier block results in Q2 having only one NOT EXISTS. Notice further that Q2 requires m tables in each FROM clause, whereas Q3 takes the intersection of m SQL queries, each with a single table in the FROM clause.

The foregoing example shows that formulas returned by algorithm NaiveFo can be "optimized" so as to have lower quantifier rank and/or less (alternations of) quantifiers blocks. The theoretical details will be given in the next section.

6 Syntactic Simplifications

Consider the second last line of algorithm NaiveFo, which specifies the first-order formula φ returned by a call $NaiveFo(q, V, E)$ with $q \neq \emptyset$. Since NaiveFo is called recursively once for each atom of q, the algorithm can return a formula with $2|q|$ quantifier blocks and with quantifier rank as high as $\mathsf{aritysum}(q)$.

In this section, we present some theoretical results that can be used for constructing "simpler" certain first-order rewritings. Section 6.1 implies that NaiveFo can be easily modified so as to return formulas with less (alternations of) quantifier blocks. Section 6.2 presents a method for decreasing the quantifier rank. Importantly, our simplifications do not decrease (nor increase) the number of $\exists$ or $\forall$ quantifiers in a formula; they merely group quantifiers of the same type in blocks and/or decrease the nesting depth of quantifiers.

6.1 Reducing the Number of Quantifier Blocks

Algorithm NaiveFo constructs a certain first-order rewriting by treating one unattacked atom at a time. The next theorem implies that multiple unattacked atoms can be "rewritten" together, which generally results in less (alternations of) quantifier blocks, as expressed by Corollary 1.

Theorem 2. *Let $Q = (q, V)$ be an acyclic* SJFCQ *query. Let $S \subseteq q$ be a set of unattacked atoms in Q's attack graph. Let $X = \left(\bigcup_{F \in S} \mathsf{KeyVars}(F)\right) \setminus V$. If φ is a certain first-order rewriting for $(q, V \cup X)$, then $\exists X \varphi$ is a certain first-order rewriting for Q.*

Corollary 1. *Let (Q, V) be an acyclic* SJFCQ *query whose attack graph is acyclic. Let p be the number of atoms on the longest directed path in the attack graph of Q. There exists a certain first-order rewriting φ for Q such that* $\mathsf{qbn}(\varphi) \leq 2p$.

Example 4. The longest path in the attack graph of Fig. 2 contains 3 atoms. From Corollary 1 and Proposition 1, it follows that the query of Fig. 2 has a certain first-order rewriting with quantifier alternation depth less than or equal to 6.

6.2 Reducing the Quantifier Rank

Consider query $\llbracket m \rrbracket$ with $m \geq 1$ in Example 3. Algorithm NaiveFo will "rewrite" the atoms $R_i(\underline{x_i}, b)$ sequentially ($1 \leq i \leq m$). However, since these atoms have no bound variables in common, it is correct to rewrite them "in parallel" and then join the resulting formulas. This idea is generalized in the following theorem.

Definition 1. *Let $Q = (q, V)$ be an* SJFCQ *query with $q \neq \emptyset$. An* independent partition *of Q is a (complete disjoint) partition $\{q_1, \ldots, q_k\}$ of q such that for $1 \leq i < j \leq k$,* $\mathsf{vars}(q_i) \cap \mathsf{vars}(q_j) \subseteq V$.

Theorem 3. *Let $Q = (q, V)$ be an acyclic* SJFCQ *query. Let $\{q_1, \ldots, q_k\}$ be an independent partition of Q. For each $1 \leq i \leq k$, let φ_i be a certain first-order rewriting for $Q_i = (q_i, V_i)$, where $V_i = V \cap \mathsf{vars}(q_i)$. Then, $\bigwedge_{i=1}^{k} \varphi_i$ is a certain first-order rewriting for Q.*

We show next that Theorem 3 gives us an upper bound on the quantifier rank of certain first-order rewritings. Intuitively, given a join tree τ, we define $\mathsf{diameter}(\tau)$ as the maximal sum of arities found on any path in τ. A formal definition follows.

Definition 2. *Let $Q = (q, V)$ be an acyclic* SJFCQ *query. Let τ be a join tree for Q. A* chain *in τ is a subset $q' \subseteq q$ such that the subgraph of τ induced by q' is a path graph. We define* $\mathsf{diameter}(\tau)$ *as the largest integer n such that* $n = \mathsf{aritysum}(q')$ *for some chain q' in τ.*

Example 5. The join tree τ in Fig. 2 (left) contains three maximal (under set inclusion) chains: $\{F, G, H\}$, $\{F, G, I\}$, $\{G, H, I\}$. The chains containing F have the greatest sum of arities; we have $\mathsf{diameter}(\tau) = 3 + 3 + 2 = 8$.

Schema	Arity	Number of tuples	Size
$\mathsf{PAGE}[\underline{\mathsf{id}}, \mathsf{namespace}, \mathsf{title}, \ldots]$	12	4,862,082	417 MB
$\mathsf{CATEGORY}[\underline{\mathsf{id}}, \mathsf{title}, \ldots]$	6	296,002	13 MB
$\mathsf{CATEGORYLINKS}[\underline{\mathsf{from}, \mathsf{to}}, \ldots]$	7	14,101,121	1.8 GB
$\mathsf{INTERWIKI}[\underline{\mathsf{prefix}}, \mathsf{url}, \ldots]$	6	662	40 KB
$\mathsf{EXTERNALLINKS}[\underline{\mathsf{from}, \mathsf{to}}, \ldots]$	3	6,933,703	1.3 GB
$\mathsf{IMAGELINKS}[\underline{\mathsf{from}, \mathsf{to}}]$	2	12,419,720	428 MB

$$Q_{4,\mathrm{fr}} = \exists^* \left[\begin{array}{l} \mathsf{PAGE}(\underline{x}, n, l, \ldots) \land \mathsf{CATEGORYLINKS}(\underline{x, t}, \ldots) \land \\ \mathsf{CATEGORY}(\underline{y}, t, \ldots) \land \mathsf{INTERWIKI}(\underline{\text{`fr'}}, u, \ldots) \end{array}\right]$$

$$Q_{5,\mathrm{fr}} = \exists^* \left[\begin{array}{l} \mathsf{PAGE}(\underline{x}, n, l, \ldots) \land \mathsf{CATEGORYLINKS}(\underline{x, t}, \ldots) \land \\ \mathsf{CATEGORY}(\underline{y}, t, \ldots) \land \mathsf{EXTERNALLINKS}(\underline{x, u}, \ldots) \land \\ \mathsf{INTERWIKI}(\underline{\text{`fr'}}, u, \ldots) \end{array}\right]$$

$$Q_{6,\mathrm{fr}} = \exists^* \left[\begin{array}{l} \mathsf{PAGE}(\underline{x}, n, l, \ldots) \land \mathsf{CATEGORYLINKS}(\underline{x, t}, \ldots) \land \\ \mathsf{CATEGORY}(\underline{y}, t, \ldots) \land \mathsf{EXTERNALLINKS}(\underline{x, u}, \ldots) \land \\ \mathsf{INTERWIKI}(\underline{\text{`fr'}}, u, \ldots) \land \mathsf{IMAGELINKS}(\underline{x, i}) \end{array}\right]$$

$$Q_{nB,\mathrm{fr}}(u) = \exists x \exists n \exists l \exists t \exists y \left[\begin{array}{l} \mathsf{PAGE}(\underline{x}, n, l, \ldots) \land \mathsf{CATEGORYLINKS}(\underline{x, t}, \ldots) \land \\ \mathsf{CATEGORY}(\underline{y}, t, \ldots) \land \mathsf{EXTERNALLINKS}(\underline{x, u}, \ldots) \land \\ \mathsf{INTERWIKI}(\underline{\text{`fr'}}, u, \ldots) \end{array}\right]$$

Fig. 4. Database schema and queries.

Corollary 2. *Let $Q = (q, V)$ be an acyclic* SJFCQ *query whose attack graph is acyclic. Let τ be a join tree for Q. There exists a certain first-order rewriting φ for Q such that* $\mathsf{qr}(\varphi) \leq \mathsf{diameter}(\tau)$.

Corollaries 1 and 2 show upper bounds on the number of quantifier blocks or the quantifier rank of certain first-order rewritings. Algorithm NaiveFo can be easily modified so as to diminish either of those measures. It is not generally possible to minimize both measures simultaneously. For example, there seems to be no certain first-order rewriting φ for $\llbracket m \rrbracket$ such that $\mathsf{qbn}(\varphi) = \mathsf{qr}(\varphi) = 2$ (cf. the table in Section 5).

7 Experiments

Databases and Queries We used a snapshot of the relational database containing Wikipedia's meta-data.[2] All experiments were conducted using MySQL version 5.1.61 on a machine with Intel core i7 2.9GHz CPU and 4GB RAM, running Gentoo Linux.

The database schema and the database size are shown in Fig. 4 (top). Attributes not shown are not relevant for our queries. To allow primary key violations, all primary key constraints and unique indexes were dropped and replaced by nonunique indexes. An inconsistent database was obtained by adding

[2] The dataset is publicly available at http://dumps.wikimedia.org/frwiki/20120117/.

	$\mathsf{qbn}(\cdot)$	$\mathsf{qr}(\cdot)$	‘fr’ variant on consistent/inconsistent database	‘zu’ variant on consistent/inconsistent database
Q_4	1	4	0.0004/0.0004	0.0004/0.0004
ν_4	7	7	0.2106/0.2114	0.2125/0.2117
β_4	3	5	$>$ 1 hour	$>$ 1 hour
ρ_4	5	4	0.2101/0.2108	0.2098/0.2106
Q_5	1	5	0.0004/0.0004	0.0004/0.0004
ν_5	9	9	0.0019/0.0020	2.1482/2.1472
β_5	5	7	0.0014/0.0015	2.1154/2.1152
ρ_5	7	6	0.0014/0.0015	2.1304/2.1298
Q_6	1	6	0.0004/0.0004	0.0004/0.0004
ν_6	11	11	0.0020/0.0021	2.1730/2.1725
β_6	5	8	0.0014/0.0015	2.1209/2.1211
ρ_6	8	6	0.0015/0.0016	2.1450/2.1439
Q_{nB}	1	4	0.0112/0.0095	0.0120/0.0099
ν_{nB}	8	8	0.0018/0.0019	2.1379/2.1374
β_{nB}	4	6	0.0015/0.0015	2.1098/2.1103
ρ_{nB}	6	5	0.0015/0.0016	2.1272/2.1274

Table 1. Execution time in seconds for eight conjunctive queries and their certain SQL rewritings according to three different rewriting algorithms: ν=naive, β reduces $\mathsf{qbn}(\cdot)$, and ρ reduces $\mathsf{qr}(\cdot)$.

$\lceil N/10000 \rceil$ conflicting tuples to each relation with N tuples. Newly added tuples are *crossovers* of existing tuples: if $R(\underline{\boldsymbol{a}_1}, \boldsymbol{b}_1)$ and $R(\underline{\boldsymbol{a}_2}, \boldsymbol{b}_2)$ are distinct tuples, then $R(\underline{\boldsymbol{a}_1}, \boldsymbol{b}_2)$ is called a crossover.

In our experiments, we used eight acyclic SJFCQ queries which are in accordance with the intended semantics of the database schema. All queries are variations of $Q_{5,\mathrm{fr}}$, which asks: “Is there some page with a category link to some category and with an external link to the wiki identified by ‘fr’?”. To help understanding, we point out that attribute CATEGORYLINKS.to refers to CATEGORY.title, not to CATEGORY.id. Four queries are shown in Fig. 4 (bottom). Each position that is not shown contains a new distinct variable not occurring elsewhere. Four other queries are obtained by replacing ‘fr’ with ‘zu’. That is, $Q_{4,\mathrm{zu}}$ is obtained from $Q_{4,\mathrm{fr}}$ by replacing ‘fr’ with ‘zu’. Likewise for $Q_{5,\mathrm{zu}}$, $Q_{6,\mathrm{zu}}$, and $Q_{nB,\mathrm{zu}}$. All queries are Boolean except for $Q_{nB,\mathrm{fr}}$ and $Q_{nB,\mathrm{zu}}$. For the Boolean queries, all ‘fr’ variants evaluate to true on the consistent database, while the ‘zu’ variants evaluate to false.

Measurements To measure the execution time $t_{Q,\mathbf{db}}$ of query Q on database $\mathbf{db}$, we execute Q ten times on $\mathbf{db}$ and take the second highest execution time. The average and standard deviation for $t_{Q,\mathbf{db}}$ are computed over 100 such measurements. Since the coefficient of variability (i.e., the ratio of standard deviation to average) was consistently less than 10^{-4} (except for the problematic queries $\beta_{4,\mathrm{fr}}$ and $\beta_{4,\mathrm{zu}}$), standard deviations have been omitted in Table 1.

In Table 1, the symbol ν refers to naive certain SQL rewritings, β refers to rewritings that decrease the number of quantifier blocks, and ρ refers to rewritings that decrease the quantifier rank. We have developed software that takes as input an acyclic SJFCQ query, tests whether it has a certain first-order rewriting, and if so, returns three certain SQL rewritings (the naive one and two simplifications). The generated SQL code is obviously awkward; for example, the SQL query $\nu_{6,\mathrm{fr}}$ contains 10 nested `NOT EXISTS`.

The values for $\mathsf{qbn}(\cdot)$ and $\mathsf{qr}(\cdot)$ were obtained by applying the definitions of Section 2 on formulas in tuple relational calculus (TRC). The motivation for this is that SQL uses aliases ranging over tables, just like TRC. The quantifier rank of a formula generally decreases when it is translated from first-order logic into TRC. For example, $\varphi = \exists x \exists y \exists z \big(R(\underline{x}, y) \wedge S(\underline{y}, z)\big)$ is translated into $\psi = \exists r \exists s \big(R(r) \wedge S(s) \wedge r.2 = s.1\big)$, where $\mathsf{qr}(\varphi) = 3$ and $\mathsf{qr}(\psi) = 2$. The quantifier rank of a Boolean SJFCQ query in TRC is equal to its number of atoms.

Observations The certain SQL rewritings of $Q_{4,\mathrm{fr}}$ (and $Q_{4,\mathrm{zu}}$) show some anomalous behavior: their execution time is much higher than rewritings of other queries involving more atoms. Since we have not been able to find the causes of this behavior in MySQL, we leave these queries out from the discussion that follows.

The execution times of the certain SQL rewritings on large databases seem acceptable in practice. A further analysis reveals the following:

- As can be expected, most conjunctive queries (indicated by Q) execute faster than their certain SQL rewritings (indicated by ν, β, ρ). The nonBoolean query $Q_{nB,\mathrm{fr}}$ is the only exception.
- In most rewritings, replacing ‘fr’ with ‘zu’ results in a considerable increase of execution time. Recall that the ‘fr’ variants of the Boolean queries evaluate to true on the consistent database, while the ‘zu’ variants evaluate to false. A possible explanation for the observed time differences is that ‘fr’ variants of Boolean queries can terminate as soon as some valuation for the tuple variables in the query makes the query true, whereas ‘zu’ variants have to range over all possible valuations.
- Execution times on consistent and inconsistent databases are almost identical. Recall that both databases only have nonunique indexes.
- Syntactic simplifications have a fairly low effect on execution times. Define the *speedup* of some simplified rewriting as the ratio of the execution time of the naive rewriting to that of the simplified one. For example, the speedup of $\rho_{5,\mathrm{fr}}$ on the consistent database is equal to $\frac{0.0019}{0.0014} \approx 1.36$. All speedups are between 1 and 1.5.

8 Conclusion

We focused on the class of acyclic SJFCQ queries that have a certain first-order rewriting. A syntactic characterization of this class is given in [15, 17]. We first implemented this earlier theory in a simple algorithm NaiveFo for constructing certain first-order rewritings. We then proposed two syntactic simplifications for such rewritings, which consist in reducing the number of quantifier blocks and reducing the quantifier rank. Our implementation indicates that certain SQL rewriting is an effective and efficient technique for obtaining certain answers. Also, it seems that naive rewritings perform well on existing database technology, and that syntactic simplifications do not result in important speedups.

References

1. M. Arenas, L. E. Bertossi, and J. Chomicki. Consistent query answers in inconsistent databases. In *PODS*, pages 68–79. ACM Press, 1999.
2. C. Beeri, R. Fagin, D. Maier, and M. Yannakakis. On the desirability of acyclic database schemes. *J. ACM*, 30(3):479–513, 1983.
3. L. E. Bertossi. *Database Repairing and Consistent Query Answering.* Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.
4. J. Chomicki and J. Marcinkowski. Minimal-change integrity maintenance using tuple deletions. *Inf. Comput.*, 197(1-2):90–121, 2005.
5. N. N. Dalvi, C. Ré, and D. Suciu. Probabilistic databases: diamonds in the dirt. *Commun. ACM*, 52(7):86–94, 2009.
6. N. N. Dalvi, C. Re, and D. Suciu. Queries and materialized views on probabilistic databases. *J. Comput. Syst. Sci.*, 77(3):473–490, 2011.
7. A. Fuxman, E. Fazli, and R. J. Miller. Conquer: Efficient management of inconsistent databases. In F. Özcan, editor, *SIGMOD Conference*, pages 155–166. ACM, 2005.
8. A. Fuxman and R. J. Miller. First-order query rewriting for inconsistent databases. *J. Comput. Syst. Sci.*, 73(4):610–635, 2007.
9. P. G. Kolaitis and E. Pema. A dichotomy in the complexity of consistent query answering for queries with two atoms. *Inf. Process. Lett.*, 112(3):77–85, 2012.
10. L. Libkin. *Elements of Finite Model Theory.* Springer, 2004.
11. D. Maslowski and J. Wijsen. A dichotomy in the complexity of counting database repairs. *J. Comput. Syst. Sci.* In press.
12. D. Maslowski and J. Wijsen. On counting database repairs. In G. H. L. Fletcher and S. Staworko, editors, *LID*, pages 15–22. ACM, 2011.
13. J. D. Ullman. *Principles of Database and Knowledge-Base Systems, Volume I.* Computer Science Press, 1988.
14. J. Wijsen. On the consistent rewriting of conjunctive queries under primary key constraints. *Inf. Syst.*, 34(7):578–601, 2009.
15. J. Wijsen. On the first-order expressibility of computing certain answers to conjunctive queries over uncertain databases. In J. Paredaens and D. V. Gucht, editors, *PODS*, pages 179–190. ACM, 2010.
16. J. Wijsen. A remark on the complexity of consistent conjunctive query answering under primary key violations. *Inf. Process. Lett.*, 110(21):950–955, 2010.
17. J. Wijsen. Certain conjunctive query answering in first-order logic. *ACM Trans. Database Syst.*, 37(2), 2012.