Minimal-Change Integrity Maintenance Using Tuple Deletions*

Jan Chomicki†
University at Buffalo
Dept. CSE
chomicki@cse.buffalo.edu

Jerzy Marcinkowski
Wroclaw University
Instytut Informatyki
jma@ii.uni.wroc.pl

30th May 2003

Abstract

We address the problem of minimal-change integrity maintenance in the context of integrity constraints in relational databases. We assume that integrity-restoration actions are limited to tuple deletions. We identify two basic computational issues: *repair checking* (is a database instance a repair of a given database?) and *consistent query answers* [ABC99] (is a tuple an answer to a given query in every repair of a given database?). We study the computational complexity of both problems, delineating the boundary between the tractable and the intractable. We consider denial constraints, general functional and inclusion dependencies, as well as key and foreign key constraints. Our results shed light on the computational feasibility of minimal-change integrity maintenance. The tractable cases should lead to practical implementations. The intractability results highlight the inherent limitations of any integrity enforcement mechanism, e.g., triggers or referential constraint actions, as a way of performing minimal-change integrity maintenance.

1 Introduction

Inconsistency is a common phenomenon in the database world today. Even though integrity constraints successfully capture data semantics, the actual data in the database often fails to satisfy such constraints. This may happen because the data is drawn from a variety of independent sources (as in data integration [Len02]) or is involved in complex, long-running activities like workflows.

How to deal with inconsistent data? The traditional way is not to allow the database to become inconsistent by aborting updates or transactions leading to integrity violations. We argue that in present-day applications this scenario is becoming increasingly impractical. First, if a violation occurs because of data from multiple, independent sources being merged [LM96], there is no single update responsible for the violation. Moreover, the updates have

*Research supported by NSF Grant IIS-0119186.

†Contact author. Address: Dept. CSE, 201 Bell Hall, Univ. at Buffalo, Buffalo, NY 14260-2000. Fax: (716) 645-3464. Phone: (716) 645-3180, ext.103.

typically already committed. For example, if we know that a person should have a single address but multiple data sources contain different addresses for the same person, it is not clear how to fix this violation through aborting some update. Second, the data may have become inconsistent through the execution of some complex activity and it is no longer possible to trace the inconsistency to a specific action.

In the context of triggers or referential integrity, more sophisticated methods for handling integrity violations have been developed. For example, instead of being aborted an update may be propagated. In general, the result is at best a consistent database state, typically with no guarantees on its distance from the original, inconsistent state (the research reported in [LML97] is an exception).

In our opinion, integrity restoration should be a separate process that is executed after an inconsistency is detected. The restoration should have a minimal impact on the database by trying to preserve as many tuples as possible. This scenario is called from now on *minimal-change integrity maintenance*.

One can interpret the postulate of minimal change in several different ways, depending on whether the information in the database is assumed to be *correct* and *complete*. If the information is complete but not necessarily correct (it may violate integrity constraints), the only way to fix the database is by *deleting* some parts of it. If the information is both incorrect and incomplete, then both insertions and deletions should be considered. In this paper we focus on the first case. Since we are working in the context of the relational data model, we consider *tuple deletions*. Such a scenario is common in data warehouse applications where dirty data coming from many sources is cleaned in order to be used as a part of the warehouse itself. On the other hand, in some data integration approaches, e.g.,[Len02, LLR02], the completeness assumption is not made. For large classes of constraints, e.g., denial constraints, the restriction to deletions has no impact, since only deletions can remove integrity violations. We return to the issue of minimal change in Section 2.

We claim that a central notion in the context of integrity restoration is that of a *repair* [ABC99]. A repair is a database instance that satisfies integrity constraints and minimally differs from the original database (which may be inconsistent). Because we consider only tuple deletions as ways to restore database consistency, the repairs in our framework are *subsets* of the original database instance.

The basic computational problem in this context is *repair checking*, namely checking whether a given database instance is a repair of of the original database. The complexity of this problem is studied in the present paper. The PTIME algorithms for repair checking given here can be easily adapted to non-deterministically compute repairs (as we show).

Sometimes when the data is obtained online from multiple, autonomous sources, it is not possible to restore the consistency. In that case one has to settle for computing, in response to queries, *consistent query answers* [ABC99], namely answers that are true in every repair of the given database. Such answers constitute a conservative "lower bound" on the information present in the database. The problem of computing consistent query answers is the second computational problem studied in the present paper. We note that the notion of consistent query answer proposed in [ABC99] has been used and extended, among others, in [ABC00, GGZ01, LLR02, ABC$^+$03, Wij03]. However, none of these papers presents a comprehensive and complete computational complexity picture.

We describe now the setting of our results. We analyze the computational complexity

of repair checking and consistent query answers along several different dimensions. We characterize the impact of the following parameters:

- the *class of queries*: quantifier-free queries, conjunctive queries, and simple conjunctive queries (conjunctive queries without repeated relation symbols).
- the *class of integrity constraints*: denial constraints, functional dependencies (FDs), inclusion dependencies (INDs), and FDs and INDs together. We also consider practically important subclasses of FDs and INDs: *key* functional dependencies and *foreign key* constraints [Dat81].
- the *number* of integrity constraints.

As a result we obtain several new classes for which both repair checking and consistent query answers are in PTIME:

- queries: ground quantifier-free, constraints: arbitrary denial;
- queries: closed simple conjunctive, constraints: functional dependencies (at most one FD per relation);
- queries: ground quantifier-free or closed simple conjunctive, constraints: key functional dependencies and foreign key constraints, with at most one key per relation.

Additionally, we show that repair checking (but not consistent query answers) are in PTIME for arbitrary FDs and acyclic INDs. The results obtained are tight in the sense that relaxing any of the above restrictions leads to co-NP-hard problems, as we prove. (This, of course, does not preclude the possibility that introducing *additional*, orthogonal restrictions could lead to more PTIME cases.) To complete the picture, we show that for arbitrary sets of FDs and INDs repair checking is co-NP-complete and consistent query answers is Π_2^p-complete.

Our results shed light on the computational feasibility of minimal-change integrity maintenance. The tractable cases should lead to practical implementations. The intractability results highlight the inherent limitations of any integrity enforcement mechanism, e.g., triggers or referential constraint actions [MS02, LML97], as ways of performing minimal-change integrity maintenance using tuple deletions.

The plan of the paper is as follows. In Section 2, we define the basic concepts. In Section 3, we consider denial constraints. In Section 4, we discuss INDs together with FDs. In Section 5, we summarize related research and in Section 6 we draw conclusions and discuss future work. An earlier version of the results in Section 3 was presented in [CM02].

2 Basic Notions

In the following we assume we have a fixed relational database schema R consisting of a finite set of relations. We also have a fixed, infinite database domain D, consisting of uninterpreted constants, and a numeric domain N. Those domains are disjoint. The database instances can be seen as finite, first-order structures over the given schema, that share the domain D. Every attribute in U is typed, thus all the instances of R can contain only elements either of D or of N in a single attribute. Since each instance is finite, it

has a finite active domain which is a subset of $D \cup N$. As usual, we allow the standard built-in predicates over N $(=, \neq, <, >, \leq, \geq)$ that have infinite, fixed extensions. With all these elements we can build a first order language $\mathcal{L}$.

2.1 Integrity Constraints

Integrity constraints are closed first-order $\mathcal{L}$-formulas. In the sequel we will denote relation symbols by $P_1, \ldots, P_m$, tuples of variables and constants by $\bar{x}_1, \ldots, \bar{x}_m$, and a conjunction of atomic formulas referring to built-in predicates by φ.

In this paper we consider the following basic classes of integrity constraints:

1. *Denial constraints*: $\mathcal{L}$-sentences

$$\forall \bar{x}_1, \ldots \bar{x}_k.\ \neg[P_1(\bar{x}_1) \wedge \cdots \wedge P_m(\bar{x}_m) \wedge \varphi(\bar{x}_1, \ldots, \bar{x}_m)].$$

2. *Functional dependencies (FDs)*: $\mathcal{L}$-sentences

$$\forall \bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 \bar{x}_5.\ [P(\bar{x}_1, \bar{x}_2, \bar{x}_4) \wedge P(\bar{x}_1, \bar{x}_3, \bar{x}_5) \Rightarrow \bar{x}_2 = \bar{x}_3],$$

   where the $\bar{x}_i$ are sequences of distinct variables. A more familiar formulation of the above FD is $X \rightarrow Y$ where X is the set of attributes of P corresponding to $\bar{x}_1$, and Y the set of attributes of P corresponding to $\bar{x}_2$ (and $\bar{x}_3$). Clearly, functional dependencies are a special case of denial constraints.

3. *Inclusion dependencies (INDs)*: $\mathcal{L}$-sentences

$$\forall \bar{x}_1\ \exists \bar{x}_3.\ [Q(\bar{x}_1) \Rightarrow P(\bar{x}_2, \bar{x}_3)],$$

   where the $\bar{x}_i$ are sequences of distinct variables with $\bar{x}_2$ contained in $\bar{x}_1$, and P, Q database relations. Again, this is often written as $Q[Y] \subseteq P[X]$ where X (resp. Y) is the set of attributes of P (resp. Q) corresponding to $\bar{x}_2$. If P and Q are clear from the context, we omit them and write the dependency simply as $Y \subseteq X$. *Full* inclusion dependencies are those expressible without the existential quantifiers.

The above constraint classes are the most common in database practice. They exhaust the constraints supported by present-day database management systems. The SQL:1999 standard [MS02] proposes general assertions that can be expressed using arbitrary SQL queries (and thus subsume arbitrary first-order constraints). However, such constraints have not found their way into practical DBMS implementations yet and are unlikely to do so in the near future. In fact, most systems allow only restricted versions of FDs and INDs in the form of key dependencies and foreign key constraints, resp.

Given a set of FDs and INDs IC and a relation P with attributes U, a *key* of P is a minimal set of attributes X of P such that IC entails the FD $X \rightarrow U$. In that case, we say that each FD $X \rightarrow Y \in IC$ is a *key* dependency and each IND $Q[Y] \subseteq P[X] \in IC$ is a *foreign key constraint*. If, additionally, X is the primary key of P, then both kinds of dependencies are termed *primary*.

Definition 1 *Given a database instance r of R and a set of integrity constraints IC, we say that r is* consistent *if $r \models IC$ in the standard model-theoretic sense;* inconsistent *otherwise.* ■

2.2 Repairs

Given a database instance r, the *set* $\Sigma(r)$ *of facts* of r is the set of ground atomic formulas $\{P(\bar{a}) \mid r \vDash P(\bar{a})\}$, where P is a relation name and $\bar{a}$ a ground tuple.

Definition 2 *The* distance $\Delta^{-}(r, r')$ *between data-base instances* r *and* r' *is defined as* $\Delta^{-}(r, r') = (\Sigma(r) - \Sigma(r'))$. ■

Definition 3 *For the instances* r, r', r'' , $r' \leq_r r''$ *if* $\Delta^{-}(r, r') \subseteq \Delta^{-}(r, r'')$, *i.e., if the distance between* r *and* r' *is less than or equal to the distance between* r *and* r''. ■

Definition 4 *Given a set of integrity constraints* IC *and database instances* r *and* r', *we say that* r' *is a* repair *of* r *w.r.t.* IC *if* $r' \vDash IC$ *and* r' *is* $\leq_r$*-minimal in the class of database instances that satisfy* IC. ■

If r' is a repair of r, then $\Sigma(r')$ is a maximal consistent subset of $\Sigma(r)$. We denote by $Repairs_{IC}(r)$ the set of repairs of r w.r.t. IC. This set is nonempty, since the empty database instance satisfies every set of FDs and INDs.

2.3 Queries

Queries are formulas over the same language $\mathcal{L}$ as the integrity constraints. A query is *closed* (or a *sentence*) if it has no free variables. A closed query without quantifiers is also called *ground*. *Conjunctive queries* [CM77, AHV95] are queries of the form

$$\exists \bar{x}_1, \ldots \bar{x}_m.\ [P_1(\bar{x}_1) \wedge \cdots \wedge P_m(\bar{x}_m) \wedge \varphi(\bar{x}_1, \ldots, \bar{x}_m)].$$

If a conjunctive query has no repeated relation symbols, it is called *simple*.

The following definition is standard:

Definition 5 *A tuple* $\bar{t}$ *is an* answer *to a query* $Q(\bar{x})$ *in an instance* r *iff* $r \models Q(\bar{t})$. ■

2.4 Consistent query answers

Given a query $Q(\bar{x})$ to r, we want as *consistent* answers those tuples that are unaffected by the violations of IC, even when r violates IC.

Definition 6 *[ABC99] A tuple* $\bar{t}$ *is a* consistent answer *to a query* $Q(\bar{x})$ *in a database instance* r *w.r.t. a set of integrity constraints* IC *iff* $\bar{t}$ *is an answer to query* $Q(\bar{x})$ *in every repair* r' *of* r *w.r.t.* IC. *An* $\mathcal{L}$*-sentence* Q *is* consistently true *in* r *w.r.t.* IC *if it is true in every repair of* r *w.r.t.* IC. *In symbols:*

$$r \models_{IC} Q(\bar{t}) \iff r' \models Q(\bar{t}) \quad \textit{for every repair } r' \textit{ of } r \textit{ w.r.t. } IC.$$

■

Note: If the set of integrity constraints IC is clear from the context, we omit it for simplicity.

2.5 Examples

Example 1 *Consider the following instance of a relation Person*

Name	*City*	*Street*
Brown	Amherst	115 Klein
Brown	Amherst	120 Maple
Green	Clarence	4000 Transit

and the functional dependency Name → City Street. Clearly, the above instance does not satisfy the dependency. There are two repairs: one is obtained by removing the first tuple, the other by removing the second. The consistent answer to the query $Person(n, c, s)$ *is just the tuple* (Green,Clarence,4000 Transit)*. On the other hand, the query* $\exists s[Person(n, c, s)]$ *has two consistent answers:* (Brown,Amherst) *and* (Green,Clarence)*. Similarly, the query*

$$Person(\text{Brown}, \text{Amherst}, 115\ \text{Klein}) \vee Person(\text{Brown}, \text{Amherst}, 120\ \text{Maple})$$

has true *as the consistent answer. Notice that for the last two queries the approach based on removing all inconsistent tuples and evaluating the original query using the remaining tuples gives different, less informative results.*

Example 2 *Consider a database with two relations Employee(SSN,Name) and Manager(SSN). There are functional dependencies* $SSN \rightarrow Name$ *and* $Name \rightarrow SSN$*, and an inclusion dependency* $Manager[SSN] \subseteq Employee[SSN]$*. The relations have the following instances:*

Employee

SSN	*Name*
123456789	*Smith*
555555555	*Jones*
555555555	*Smith*

Manager

SSN
123456789
555555555

The instances do not violate the IND but violate both FDs. If we consider only the FDs, there are two repairs: one obtained by removing the third tuple from Employee, and the other by removing the first two tuples from the same relation. However, the second repair violates the IND. This can be fixed by removing the first tuple from Manager. So if we consider all the constraints, there are two repairs:

Employee

SSN	*Name*
123456789	*Smith*
555555555	*Jones*

Manager

SSN
123456789
555555555

and

Employee

SSN	*Name*
555555555	*Smith*

Manager

SSN
555555555

■

Example 3 *We give here some examples of denial constraints. Consider the relation Emp with attributes Name, Salary, and Manager, with Name being the primary key. The constraint that no employee can have a salary greater that that of her manager is a denial constraint:*

$$\forall n, s, m, s', m'.\ \neg[Emp(n, s, m) \wedge Emp(m, s', m') \wedge s > s'].$$

Similarly, single-tuple constraints (`CHECK` *constraints in SQL2) are a special case of denial constraints. For example, the constraint that* no employee can have a salary over $200000 *is expressed as:*

$$\forall n, s, m.\ \neg[Emp(n, s, m) \wedge s > 200000].$$

Note that a single-tuple constraint always leads to a single repair which consists of all the tuples of the original instance that satisfy the constraint.

2.6 Different notions of repair

The original notion of repair introduced in [ABC99] required that the *symmetric* difference between a database and its repair be minimized. As explained in the introduction, this was based on the assumption that the database may be not only inconsistent but also incomplete. The notion of repair pursued in the current paper (Definition 4) reflects the assumption that the database is complete. There are several reasons for this change of perspective. First, for denial constraints integrity violations can only be removed by deleting tuples, so the different notions of repair in fact coincide in this case. Therefore, all the results presented in Section 3 are not affected by the restriction of the repairs to be subsets of the original instance. Insertions can restore integrity only for inclusion dependencies (or, in general for tuple-generating dependencies [AHV95]). Second, even for inclusion dependencies current language standards like SQL:1999 [MS02] allow only deletions in their repertoire of referential integrity actions. Third, disallowing insertions significantly strengthens the notion of consistent query answer, as demonstrated by the following example.

Example 4 *Consider a database schema consisting of two relations $P(AB)$ and $S(C)$. The integrity constraints are: the FD $A \rightarrow B$ and the IND $B \subseteq C$. Assume the database instance r_1 consists of $p = \{(a, b), (a, c)\}$ and $s = \{b\}$. Then under Definition 4 there is only one repair r_2 consisting of $p' = \{(a, b)\}$ and $s' = s$. On the other hand, under the definition of [ABC99], there is one more repair r_3 consisting of $p'' = \{(a, c)\}$ and $s'' = \{b, c\}$. Therefore, in the first case $P(a, b)$ is consistently true in the original instance r_1, while in the second case it is not. Note that $P(a, c)$ is not consistently true in r_1 either. Thus, in the second case $P(a, b)$ and $P(a, c)$ are treated symmetrically from the point of view of consistent query answering. However, intuitively there is a difference between them. Think of A being the person's name, B her address and S a list of valid addresses. Then only under Definition 4 would the single valid address be returned as a consistent answer.*

Finally, insertions may lead to infinitely many repairs which are, moreover, not very intuitive as ways of fixing an inconsistent database.

Example 5 *In Example 2, allowing insertions gives additionally infinitely many repairs of the form*

Employee

SSN	*Name*
123456789	*c*
555555555	*Smith*

Manager

SSN
123456789
555555555

where c is an arbitrary string different from Smith.

2.7 Computational Problems

Assume a class of databases $\mathcal{D}$, a class of queries $\mathcal{Q}$ and a class of integrity constraints $\mathcal{C}$ are given. We study here the complexity of the following problems:

- *repair checking*, i.e., the complexity of the set

$$B_{IC} = \{(r, r') : r, r' \in \mathcal{D} \wedge r' \in \mathit{Repairs}_{IC}(r)\},$$

- *consistent query answers*, i.e., the complexity of the set

$$D_{IC,\Phi} = \{r : r \in \mathcal{D} \wedge r \models_{IC} \Phi\},$$

for a fixed sentence $\Phi \in \mathcal{Q}$ and a fixed finite set $IC \in \mathcal{C}$ of integrity constraints. This formulation is called *data complexity* [CH80, Var82], since it captures the complexity of a problem as a function of the number of tuples in the database instance only. The database schema, the query and the integrity constraints are assumed to be fixed.

It is easy to see that even under a single key FD, there may be exponentially many repairs and thus the approach to computing consistent query answers by generating and examining all repairs is not feasible.

Example 6 *Consider the functional dependency $A \rightarrow B$ and the following family of relation instances r_n, $n > 0$, each of which has $2n$ tuples (represented as columns) and 2^n repairs:*

r_n							
A	a_1	a_1	a_2	a_2	$\cdots$	a_n	a_n
B	b_0	b_1	b_0	b_1	$\cdots$	b_0	b_1

We establish below a general relationship between the problems of repair checking and consistent query answers.

Theorem 1 *In the presence of foreign key constraints, the problem of repair checking is logspace-reducible to the complement of the problem of consistent query answers.*

Proof. We discuss here the case of the database consisting of a single relation R_0. Assume r is the given instance of R_0 and r' is an another instance of R_0 satisfying the set of integrity constraints IC. We define a new relation S_0 having the same attributes as R_0 plus an additional attribute Z. Consider an instance s of S_0 built as follows:

- for every tuple $(x_1, \ldots, x_k) \in r'$, we add the tuple $(x_1, \ldots, x_k, c_1)$ to s;
- for every tuple $(x_1, \ldots, x_k) \in r - r'$, we add the tuple $(x_1, \ldots, x_k, c_2)$ to s.

Consider also another relation P having a single attribute W , and a foreign key constraint $i_0 : P[W] \subseteq S_0[Z]$. The instance p of P consists of a single tuple c_2. We claim that $P(c_2)$ is consistently true in the database instance consisting of s and p w.r.t. $IC \cup \{i_0\}$ iff r' is not a repair of r w.r.t. IC. ∎

3 Denial constraints

3.1 Conflict hypergraph

Given a set of denial constraints F and an instance r, all the repairs of r with respect to F can be succinctly represented as the *conflict hypergraph*. This is a generalization of the *conflict graph* defined in [ABC01] for FDs only.

Definition 7 *The* conflict hypergraph $\mathcal{G}_{F,r}$ *is a hypergraph whose set of vertices is the set* $\Sigma(r)$ *of facts of an instance* r *and whose set of edges consists of all the sets*

$$\{P_1(\bar{t}_1), P_2(\bar{t}_2), \ldots P_l(\bar{t}_l)\}$$

such that $P_1(\bar{t}_1), P_2(\bar{t}_2), \ldots P_l(\bar{t}_l) \in \Sigma(r)$*, and there is a constraint*

$$\forall \bar{x}_1, \bar{x}_2, \ldots \bar{x}_l.\ \neg[P_1(\bar{x}_1) \wedge P_2(\bar{x}_2) \wedge \ldots \wedge P_l(\bar{x}_l) \wedge \varphi(\bar{x}_1, \bar{x}_2, \ldots \bar{x}_l)]$$

in F *such that* $P_1(\bar{t}_1), P_2(\bar{t}_2), \ldots P_l(\bar{t}_l)$ *violate together this constraint, which means that there exists a substitution* ρ *such that* $\rho(\bar{x}_1) = \bar{t}_1, \rho(\bar{x}_2) = \bar{t}_2, \ldots \rho(\bar{x}_l) = \bar{t}_l$ *and that* $\varphi(\bar{t}_1, \bar{t}_2, \ldots \bar{t}_l)$ *is true.*

Note that there may be edges in $\mathcal{G}_{F,r}$ that contain only one vertex. Also, the size of the conflict hypergraph is polynomial in the number of tuples in the database instance.

By an *independent set* in a hypergraph we mean a subset of its set of vertices which does not contain any edge.

Proposition 1 *Each repair of* r *w.r.t.* F *corresponds to a maximal independent set in* $\mathcal{G}_{F,r}$*.*

Proposition 1 yields the following result:

Proposition 2 *[ABC$^+$03] For every set of denial constraints* F *and* $\mathcal{L}$*-sentence* Φ*,* B_F *is in PTIME and* $D_{F,\Phi}$ *is in co-NP.* ■

Note that the repairs of an instance r can be computed nondeterministically by picking a vertex of $\mathcal{G}_{F,r}$ which does not belong to a single-vertex edge and adding vertices that do not result in the addition of an entire edge.

3.2 Positive results

A set of constraints is *generic* if it does not imply any ground literal. The results in [ABC99] imply the following:

Proposition 3 *For every generic set* F *of binary denial constraints and full inclusion dependencies, and quantifier-free* $\mathcal{L}$*-sentence*

$$\Phi = P_1(\bar{x}_1) \wedge \cdots P_m(\bar{x}_m) \wedge \neg P_{m+1}(\bar{x}_{m+1}) \wedge \cdots \wedge \neg P_n(\bar{x}_n) \wedge \varphi(\bar{x}_1, \ldots, \bar{x}_n),$$

$D_{F,\Phi}$ *is in PTIME.* ■

The techniques in [ABC99] do not generalize to non-binary constraints, or queries involving disjunction or quantifiers. However, non-binary constraints and disjunctions do not necessarily lead to intractability, as shown by the following theorem.

Theorem 2 *For every set F of denial constraints and quantifier-free $\mathcal{L}$-sentence Φ, $D_{F,\Phi}$ is in PTIME.* ■

Proof. We assume the sentence is in CNF, i.e., of the form $\Phi = \Phi_1 \wedge \Phi_2 \wedge \ldots \Phi_l$, where each Φ_i is a disjunction of ground literals. Φ is true in every repair of r if and only if each of the clauses Φ_i is true in every repair. So it is enough to provide a polynomial algorithm which will check if a given ground clause is consistently *true*.

It is easier to think that we are checking if a ground clause *true* is **not** consistently true. This means that we are checking, whether there exists a repair r' in which $\neg\Phi_i$ is true for some i. But $\neg\Phi_i$ is of the form $P_1(\bar{t}_1) \wedge P_2(\bar{t}_2) \wedge \ldots \wedge P_m(\bar{t}_m) \wedge \neg P_{m+1}(\bar{t}_{m+1}) \wedge \ldots \wedge \neg P_n(\bar{t}_n)$, where the $\bar{t}_j$'s are tuples of constants. WLOG, we assume that all the facts in the set $\{P_1(\bar{t}_1), \ldots, P_n(\bar{t}_n))\}$ are mutually distinct.

The nonderministic algorithm selects for every j, $m+1 \leq j \leq n$, $P_j(\bar{t}_j) \in \Sigma(r)$, an edge $E_j \in \mathcal{G}_{F,r}$ such that $P_j(\bar{t}_j) \in E_j$, and constructs a set of facts S such that

$$S = \{P_1(\bar{t}_1), \ldots, P_m(\bar{t}_m)\} \cup \bigcup_{m+1 \leq j \leq n, P_j(\bar{t}_j) \in \Sigma(r)} (E_j - \{P_j(\bar{t}_j)\})$$

and *there is no edge $E \in \mathcal{G}_{F,r}$ such that $E \subseteq S$*. If the construction of S succeeds, then a repair in which $\neg\Phi_i$ is true can be built by adding to S new facts from $\Sigma(r)$ until the set is maximal independent. The algorithm needs $n - m$ nondeterministic steps, a number which is independent of the size of the database (but dependent on Φ), and in each of its nondeterministic steps selects one possibility from a set whose size is polynomial in the size of the database. So there is an equivalent PTIME deterministic algorithm. ■

In the case when the set F of integrity constraints consists of only one FD per relation the conflict hypergraph has a very simple form. It is a disjoint union of full multipartite graphs. If this single dependency is a key dependency then the conflict graph is a union of disjoint cliques. Because of this very simple structure we hoped that it would be possible, in such a situation, to compute in polynomial time the consistent answers not only to ground queries, but also to all conjunctive queries. As we are going to see now, this is only possibly if the conjunctive queries are suitably restricted.

Theorem 3 *Let F be a set of FDs, each dependency over a different relation among $P_1, P_2, \ldots, P_k$. Then for each closed simple conjunctive query Q, there exists a sentence Q' such that for every database instance r, $r \models_F Q$ iff $r \models Q'$. Consequently, $D_{F,Q}$ is in PTIME.*

Proof. We present the construction for $k = 2$ for simplicity; the generalization to an arbitrary k is straightforward. Let P_1 and P_2 be two different relations of arity k_1 and k_2, resp. Assume we have the following FDs: $Y_1 \rightarrow Z_1$ over P_1 and $Y_2 \rightarrow Z_2$ over P_2. Let $\bar{y}_1$ be a vector of arity $|Y_1|$, $\bar{y}_2$ a vector of arity $|Y_2|$, $\bar{z}_1$ and $\bar{z}'_1$ vectors of arity $|Z_1|$, and $\bar{z}_2$ and $\bar{z}'_2$ vectors of arity $|Z_2|$. Finally, let $\bar{w}_1, \bar{w}'_1, \bar{w}''_1$ (resp. $\bar{w}_2, \bar{w}'_2, \bar{w}''_2$) be vectors of arity

$k_1 - |Y_1| - |Z_1|$ (resp. $k_2 - |Y_2| - |Z_2|$). All of the above vectors consist of distinct variables. The query Q is of the following form

$$\exists \bar{y}_1, \bar{z}_1, \bar{w}_1, \bar{y}_2, \bar{z}_2, \bar{w}_2.\ [P_1(\bar{y}_1, \bar{z}_1, \bar{w}_1) \wedge P_2(\bar{y}_2, \bar{z}_2, \bar{w}_2) \wedge \varphi(\bar{y}_1, \bar{z}_1, \bar{w}_1, \bar{y}_2, \bar{z}_2, \bar{w}_2)].$$

Then, the query Q' is as follows:

$$\exists \bar{y}_1, \bar{z}_1, \bar{w}_1, \bar{y}_2, \bar{z}_2, \bar{w}_2 \forall \bar{z}'_1, \bar{w}'_1, \bar{z}'_2, \bar{w}'_2 \exists \bar{w}''_1, \bar{w}''_2 [P_1(\bar{y}_1, \bar{z}_1, \bar{w}_1) \wedge P_2(\bar{y}_2, \bar{z}_2, \bar{w}_2) \wedge \varphi(\bar{y}_1, \bar{z}_1, \bar{w}_1, \bar{y}_2, \bar{z}_2, \bar{w}_2)$$
$$\wedge (P_1(\bar{y}_1, \bar{z}'_1, \bar{w}'_1) \wedge P_2(\bar{y}_2, \bar{z}'_2, \bar{w}'_2) \Rightarrow P_1(\bar{y}_1, \bar{z}'_1, \bar{w}''_1) \wedge P_2(\bar{y}_2, \bar{z}'_2, \bar{w}''_2) \wedge \varphi(\bar{y}_1, \bar{z}'_1, \bar{w}''_1, \bar{y}_2, \bar{z}'_2, \bar{w}''_2))].$$

■

We show now that the above results are the strongest possible, since relaxing any of the restrictions leads to co-NP-completeness. This is the case even though we limit ourselves to *key* FDs.

3.3 One key dependency, nonsimple conjunctive query

Theorem 4 *There exist a key FD f and a closed conjunctive query*

$$Q \equiv \exists x, y, z.\ [R(x, y, c) \wedge R(z, y, c')],$$

for which $D_{\{f\},Q}$ is co-NP-complete.

Proof. Reduction from MONOTONE 3-SAT. The FD is $A \to BC$. Let $\Phi = \phi_1 \wedge \ldots \phi_m \wedge \psi_{m+1} \ldots \wedge \psi_l$ be a conjunction of clauses, such that all occurrences of variables in ϕ_i are positive and all occurrences of variables in ψ_i are negative. We build a database with the facts $R(i, p, c)$ if the variable p occurs in the clause ψ_i and $R(i, p, c')$ if the variable p occurs in the clause ϕ_i. Now, there is an assignment which satisfies Φ if and only if there exists a repair of the database in which Q is false. To show the $\Rightarrow$ implication, select for each clause ϕ_i one variable p_i which occurs in this clause and whose value is 1 and for each clause ψ_i one variable p_i which occurs in ψ_i and whose value is 0. The set of facts $\{R(i, p_i, c) : i \leq m\} \cup \{R(i, p_i, c') : m + 1 \leq i \leq l\}$ is a repair in which the query Q is false. The $\Leftarrow$ implication is even simpler. ■

3.4 Two key dependencies, single-atom query

By a *bipartite edge-colored graph* we mean a tuple $\mathcal{G} = \langle V, E, B, G \rangle$ such that $\langle V, E \rangle$ is an undirected bipartite graph and $E = B \cup G$ for some given disjoint sets B, G (so we think that each of the edges of $\mathcal{G}$ has one of the two colors).

Definition 8 *Let $\mathcal{G} = \langle V, E, B, G \rangle$ be a bipartite edge-colored graph, and let $M \subset E$. We say that M is maximal $\mathcal{V}$-free if:*

1. *M is a maximal (w.r.t. inclusion) subset of E with the property that neither $M(x, y) \wedge M(x, z)$ nor $M(x, y) \wedge M(z, y)$ holds for any x, y, z.*
2. *$M \cap B = \emptyset$.*

We say that $\mathcal{G}$ has the max-$\mathcal{V}$-free property if there exists M which is maximal $\mathcal{V}$-free.

Lemma 1 *Max-$\mathcal{V}$-free is an NP-complete property of bipartite edge-colored graphs.*

Proof. Reduction from 3-COLORABILITY. Let $\mathcal{H} = \langle U, D\rangle$ be some undirected graph. This is how we define the bipartite edge-colored graph $\mathcal{G}_{\mathcal{H}}$:

1. $V = \{v_\varepsilon, v'_\varepsilon : v \in U, \varepsilon \in \{m, n, r, g, b\}\}$, which means that there are 10 nodes in the graph $\mathcal{G}$ for each node of $\mathcal{H}$;
2. $G(v_m, v'_r), G(v_m, v'_b), G(v_n, v'_b), G(v_n, v'_g)$ and $G(v_r, v'_m)$,$G(v_b, v'_m)$,$G(v_b, v'_n)$,$G(v_g, v'_n)$ hold for each $v \in U$;
3. $B(v_\epsilon, v'_\varepsilon)$ holds for each $v \in U$ and each pair $\epsilon, \varepsilon \in \{r, g, b\}$ such that $\epsilon \neq \varepsilon$;
4. $B(v_\varepsilon, u'_\varepsilon)$ holds for each $\varepsilon \in \{r, g, b\}$ and each pair $u, v \in U$ such that $D(u, v)$.

Suppose that $\mathcal{H}$ is 3-colorable. We fix a coloring of $\mathcal{H}$ and construct the set M. For each $v \in U$: if the color of v is Red, then the edges $G(v_m, v'_b), G(v_n, v'_g)$ and $G(v_b, v'_m), G(v_g, v'_n)$ are in M. If color of v is Green, then the edges $G(v_m, v'_r), G(v_n, v'_b)$ and $G(v_r, v'_m), G(v_b, v'_n)$ are in M, and if the color of v is Blue, then the edges $G(v_m, v'_r), G(v_n, v'_g)$ and $G(v_r, v'_m), G(v_g, v'_n)$ are in M. It is easy to see that the set M constructed in this way is maximal $\mathcal{V}$-free.

For the other direction, suppose that a maximal $\mathcal{V}$-free set M exists in $\mathcal{G}_{\mathcal{H}}$. Then, for each $v \in U$ there is at least one node among v_r, v_g, v_b which does not belong to any G-edge in M. Let v_ϵ be this node. Also, there is at least one such node (say, v'_ε) among v'_r, v'_g, v'_b. Now, it follows easily from the construction of $\mathcal{G}_{\mathcal{H}}$ that if M is maximal $\mathcal{V}$-free then $\epsilon = \varepsilon$. Let this ϵ be color of v in $\mathcal{G}$. It is easy to check that the coloring defined in this way is a legal 3-coloring of $\mathcal{G}$. ■

Theorem 5 *There is a set F of two key dependencies and a closed conjunctive query $Q \equiv \exists x, y.\ [R(x, y, b)]$, for which $D_{F,Q}$ is co-NP-complete.*

Proof. The 2 dependencies are $A \to BC$ and $B \to AC$. For a given bipartite edge-colored graph $\mathcal{G} = \langle V, E, B, G\rangle$ we build a database with the tuples (x, y, g) if $G(x, y)$ holds in $\mathcal{G}$ and (x, y, b) if $B(x, y)$ holds in $\mathcal{G}$. Now the theorem follows from Lemma 1 since a repair in which the query Q is not true exists if and only if $\mathcal{G}$ has the max-$\mathcal{V}$-free property. ■

3.5 One denial constraint

By an *edge-colored graph* we mean a tuple $\mathcal{G} = \langle V, E, P, G, B\rangle$ such that $\langle V, E\rangle$ is a (directed) graph and $E = P \cup G \cup B$ for some given pairwise disjoint sets P, G, B (which we interpret as colors). We say that the edge colored graph $\mathcal{G}$ has the $\mathcal{Y}$ property if there are $x, y, z, t \in E$ such that $E(x, y), E(y, z), E(y, t)$ hold and the edges $E(y, z)$ and $E(y, t)$ are of different colors.

Definition 9 *We say that the edge-colored graph $\langle V, E, P, G, B\rangle$ has the max-$\mathcal{Y}$-free property if there exists a subset M of E such that $M \cap P = \emptyset$ and :*

1. *$\langle V, M, P \cap M, G \cap M, B \cap M\rangle$ does not have the $\mathcal{Y}$-property;*

2. M is a maximal (w.r.t. inclusion) subset of E satisfying the first condition;

Lemma 2 *Max-$\mathcal{Y}$-free is an NP-complete property of edge-colored graphs.*

Proof. By a reduction of 3SAT. Let $\Phi = \phi_1 \wedge \phi_2 \wedge \ldots \wedge \phi_l$ be conjunction of clauses. Let $p_1, p_2, \ldots p_n$ be all the variables in Φ. This is how we define the *edge-colored graph* $\mathcal{G}_\Phi$:

1. $V = \{a_i, b_i, c_i, d_i : 1 \leq i \leq n\} \cup \{e_i, f_i, g_i : 1 \leq i \leq l\}$, which means that there are 3 nodes in the new graph for each clause in Φ and 4 nodes for each variable.
2. $P(a_i, b_i)$ and $P(e_j, f_j)$ hold for each suitable i, j;
3. $G(b_i, d_i)$ and $G(e_j, g_j)$ hold for each suitable i, j;
4. $B(b_i, c_i)$ holds for each suitable i;
5. $G(d_i, e_j)$ holds if p_i occurs positively in ϕ_j;
6. $B(d_i, e_j)$ holds if p_i occurs negatively in ϕ_j;
7. $E = B \cup G \cup P$.

Now suppose that Φ is satisfiable, and that μ is the satisfying assignment. We define the set $M \subset E$ as follows. We keep in M all the G-colored edges from item 3 above. If $\mu(p_i) = 1$ then we keep in M all the G edges leaving d_i (item 5). Otherwise we keep in M all the B edges leaving d_i (item 6). Obviously, $M \cap P = \emptyset$. It is also easy to see that M does not have the $\mathcal{Y}$-property and that it is maximal.

In the opposite direction, notice that if an M, as in Definition 9 does exist, then it must contain all the G-edges from item 2 above - otherwise a P edge could be added without leading to the $\mathcal{Y}$-property. But this means that, for each i, M can either contain some (or all) of the B-edges leaving d_i or some (or all) of the G-edges. In this sense M defines a valuation of variables. Also, if M is maximal, it must contain, for each j, at least one edge leading to e_j. But this means that the defined valuation satisfies Φ. ■

Theorem 6 *There exist a denial constraint f and a closed conjunctive query*

$$Q \equiv \exists x, y.\ [R(x, y, p)],$$

for which $D_{\{f\},Q}$ is co-NP-complete.

Proof. The denial constraint f is:

$$\forall x, y, z, s, s', s''\ \neg[R(x, y, s) \wedge R(y, z.s') \wedge R(y, w, s'') \wedge s' \neq s'']$$

For a given edge-colored graph $\mathcal{G} = \langle V, E, P, G, B\rangle$ we build a database with the tuples $R(x, y, g)$ if $G(x, y)$ holds in $\mathcal{G}$, with $R(x, y, p)$ if $P(x, y)$ holds in $\mathcal{G}$ and with $R(x, y, b)$ if $B(x, y)$ holds in $\mathcal{G}$. Now the theorem follows from Lemma 2 since a repair in which the query Q is not true exists iff $\mathcal{G}$ has the max-$\mathcal{Y}$-free property. ■

4 Inclusion dependencies

Proposition 4 *For every set of INDs I and $\mathcal{L}$-sentence Φ, B_I and $D_{I,\Phi}$ are in PTIME.*

Proof. For a given database instance r, a single repair is obtained by deleting all the tuples violating I (and only those). ■

We consider now FDs and INDs together.

4.1 Single-key relations

We want to identify here the cases where both repair checking and computing consistent query answers can be done in PTIME. The intuition is to limit the interaction between the FDs and the INDs in the given set of integrity constraints in such a way that one can use the PTIME results obtained for FDs in the previous section and in [ABC$^+$03].

Lemma 3 *Let $IC = F \cup I$ be a set of constraints consisting of a set of key FDs F and a set of foreign key constraints I but with no more than one key per relation. Let r be a database instance and r' be the unique repair of r with respect to the foreign key constraints in I. Then r'' is a repair of r w.r.t. IC if and only if it is a repair of r' w.r.t. F.*

Proof. The only thing to be noticed here is that repairing r' with respect to key constraints does not lead to new inclusion violations. This is because the set of key values in each relation remains unchanged after such a repair (which is not necessarily the case if we have relations with more than one key). ■

Corollary 1 *Under the assumptions of Lemma 3, B_{IC} is in PTIME.*

Proof. Follows from Proposition 2. ■

The repairs w.r.t. $IC = F \cup I$ of r are computed by (deterministically) repairing r w.r.t. I and then nondeterministically repairing the result w.r.t. F (as described in the previous section).

We can also transfer the PTIME results about consistent query answers obtained for FDs only.

Corollary 2 *Let Φ a quantifier-free $\mathcal{L}$-sentence or a simple conjunctive closed $\mathcal{L}$-query. Then under the assumptions of Lemma 3, $D_{IC,\Phi}$ is in PTIME.*

Proof. From Theorem 2 and Theorem 3. ■

Unfortunately, the cases identified above are the only ones we know of in which both repair checking and consistent query answers are in PTIME.

4.2 Acyclic inclusion dependencies

For acyclic INDs (and arbitrary FDs), the repair checking problem is still in PTIME. Surprisingly, consistent query answers becomes in this case a co-NP-hard problem, even in the case of key FDs and primary key foreign key constraints. If we relax any of the assumptions of Lemma 3, the problem of consistent query answers becomes intractable, even under acyclicity.

Definition 10 *[AHV95] Let I be a set of INDs over a database schema R. Consider a directed graph whose vertices are relations from R and such that there is an edge $E(P, R)$ in the graph if and only if there is an IND of the form $P[X] \subseteq R[Y]$ in I. A set of inclusion dependencies is* acyclic *if the above graph does not have a cycle.* ■

Theorem 7 *Let $IC = F \cup I$ be a set of constraints consisting of a set of FDs F and an acyclic set of INDs I. Then B_{IC} is in PTIME.*

Proof. First compare r and r' on relations which are not on the left-hand side of any IND in I. Here, r' is a repair if and only if the functional dependencies are satisfied in r' and if adding to it any additional tuple from r would violate one of the functional dependencies. Then consider relations which are on the left-hand side of some INDs, but the inclusions only lead to already checked relations. Again, r' is a repair of those relations if and only if adding any new tuple (i.e. any tuple from r but not from r') would violate some constraints. Repeat the last step until all the relations are checked. ■

The above proof yields a nondeterministic PTIME procedure for computing the repairs w.r.t. $IC = F \cup I$.

To our surprise, Theorem 7 is the strongest possible positive result. The problem of consistent query answers is already intractable, even under additional restrictions on the FDs and INDs. To see this let us start by establishing NP-completeness of the *maximal spoiled-free* problem.

By an instance of the maximal spoiled-free problem we will mean $\mathcal{G} = \langle V, V_1, V_2, V_3, S, E\rangle$ such that:

1. $\langle V, E\rangle$ is a ternary undirected hypergraph (so V is a set of vertices and E is a set of *triangles*);
2. V_1, V_2, V_3 are pairwise disjoint;
3. $V_1 \cup V_2 \cup V_3 = V$;
4. Relation E is typed: if $E(a, b, c)$ holds in $\mathcal{G}$ then $a \in V_1$, $b \in V_2$ and $c \in V_3$;
5. $S \subseteq V_1$ (S will be called set of *spoiled vertices*).

We will consider maximal (with respect to inclusion) sets of disjoint triangles in $\mathcal{G}$. We call a triangle *spoiled* if one of its vertices is spoiled. The *maximal spoiled-free* problem is defined as the problem of deciding, for a given instance $\mathcal{G} = \langle V, V_1, V_2, V_3, S, E\rangle$, if there exists a maximal set $T \subset E$ of disjoint triangles, such that none of the triangles in T is spoiled. It is easy to get confused here, so let us explain that the problem we are considering here is not the existence of a set of disjoint triangles, which would be maximal in the class of sets not containing a spoiled triangle: such a set of course always exists. The problem we consider is the existence of a set of disjoint triangles in $\mathcal{G}$ which is not only maximal but also does not contain a spoiled triangle.

Lemma 4 *The maximal spoiled-free problem is NP-complete.*

Proof. By a reduction of 3-colorability. Let $\mathcal{H} = \langle U, D \rangle$ be some undirected graph. We are going to construct the instance of the maximal spoiled-free problem $\mathcal{G}_{\mathcal{H}}$. The construction is a little bit complicated, and we hope to simplify the presentation by the following convention:

Each vertex in V_1 belongs to exactly one triangle in E. So a triangle is fully specified by its vertex in V_2, its vertex in V_3 and by the information if it is spoiled or not.

Now, for each vertex v in U we will have vertices v_r, v_g, v_b, v_p, v_q in V_2 and vertices $v'_r, v'_g, v'_b, v'_p, v'_q$ in V_3. The only nonspoiled triangles will be the defined by the following pairs: $[v_r, v'_p]$, $[v_g, v'_p]$, $[v_g, v'_q]$, $[v_b, v'_q]$, $[v_p, v'_r]$, $[v_p, v'_g]$, $[v_q, v'_g]$, $[v_q, v'_b]$ (so we have 8 nonspoiled triangles for each vertex in U).

There are two kinds of spoiled triangles. For each $v \in U$, and for each pair $\epsilon, \varepsilon \in \{r, g, b\}$ such that $\epsilon \neq \varepsilon$ there is a spoiled triangle $[v_\epsilon, v'_\varepsilon]$ in $\mathcal{G}$. For each $v, u \in U$, such that $D(v, u)$ holds in $\mathcal{H}$, and for each $\epsilon \in \{r, g, b\}$ there is a spoiled triangle $[v_\epsilon, u_\epsilon]$ in $\mathcal{G}$.

Now we need to show that $\mathcal{H}$ is 3-colorable if and only if there exists a maximal set $T \subset E$ of disjoint triangles, such that none of the triangles in T is spoiled.

Let us start from the $\Rightarrow$ direction, which is simple. Consider a coloring of $\mathcal{H}$ with colors r, g and b. Now take T as a set containing, for each vertex v of $\mathcal{H}$ with some color ϵ, all nonspoiled triangles of the form $[v_\alpha, v'_\beta]$ where neither α nor β equals to ϵ. Obviously, T defined in this way, does not contain spoiled triangles. A simple analysis shows that it is also maximal.

For the other direction suppose that there is a set T of disjoint triangles in $\mathcal{G}$ which is maximal and only contains nonspoiled triangles. It is easy to see that for each v exactly one of the vertices v_r, v_g, v_b is not in any triangle in T, and that also among v'_r, v'_g, v'_b there is exactly one which is not in any triangle in T. If they were different, in the sense that first of them were v_ε and the second v'_ϵ, for $\epsilon \neq \varepsilon$, then a spoiled triangle $[v_\varepsilon, v'_\epsilon]$ could be added to T what contradicts its maximality. So they are equal, and in a natural way they define a color of v. Now we need to prove that the coloring of $\mathcal{H}$ defined in this way is a legal one. But if $D(u, v)$ holds in $\mathcal{H}$ then there is spoiled triangle $[v_\epsilon, u_\epsilon]$ in $\mathcal{G}$ for each $\epsilon \in \{r, g, b\}$. So if the colors of v and u were both equal to some ϵ, then we could add this spoiled triangle, and T would not be maximal. ∎

Theorem 8 *There exist a database schema, a set IC of integrity constraints consisting of key FDs and of an acyclic set of primary foreign key constraints, and a ground atomic query Φ such that $D_{IC,\Phi}$ is co-NP-hard.*

Proof. The schema consists of a unary relation P, a binary relation $Q(Q_1, Q_2)$ and of a ternary relation $R(R_1, R_2, R_3)$. The columns Q_1,R_1,R_2,R_3 are keys, with Q_1 and R_1 being the primary keys. The foreign key dependencies are $P \subseteq Q_1$ and $Q_2 \subseteq R_1$. For a given instance $\mathcal{G}$ of the maximal spoiled-free problem we will construct a database instance r, and a query Φ such that $\mathcal{G}$ has the maximal spoiled-free property if and only if there is a repair r' of r with respect to IC such that Φ is not true in r'.

We define the relation P as a single fact $P(a)$. The relation Q is defined as a set of facts $\{Q(a, s) : s \in S\}$, where S is the set of spoiled vertices from $\mathcal{G}$. Finally, R is the hypergraph from $\mathcal{G}$. The query Φ is $P(a)$.

The repairs of R with respect to the key dependencies correspond to maximal sets of disjoint triangles in $\mathcal{G}$. If $\mathcal{G}$ has the maximal spoiled-free property then there exists a repair

of R which does not contain any tuple of the form $R(s, u, v)$ with $s \in S$. But then the only way to repair Q is it take the empty relation, and, consequently, the only way to repair P is to take the empty relation. So if $\mathcal{G}$ has the maximal spoiled-free property then Φ indeed is not true in all repairs. For the other direction notice that if if each repair of R it is a tuple of the form $R(s, u, v)$ with $s \in S$ then each repair of Q is nonempty and in consequence each repair of P consists of the single atom $P(a)$, so then Φ is indeed true in all repairs. ■

4.3 Relaxing acyclicity

We show here that relaxing the acyclicity assumption in Theorem 7 leads to the intractability of the repair checking problem (and thus also the problem of consistent query answers), even though alternative restrictions on the integrity constraints are imposed.

4.3.1 One FD, one IND

Theorem 9 *There exist a database schema and a set IC of integrity constraints, consisting of one FD and one IND, such that B_{IC} is co-NP-hard.*

Proof. We will check here whether the empty set is a repair. The database schema consists of one relation $R(A_1, A_2, A_3, A_4)$ and the constraints in IC are $A_1 \to A_2$ and $A_3 \subseteq A_4$.

Consider a propositional formula $\Phi = \phi_1 \wedge \phi_2 \wedge \ldots \phi_m$, where ϕ_i are clauses. Let r_Φ consist of the facts $R(p_j, 0, \phi_i, \phi_{i+1})$ such that p_j occurs negatively in ϕ_i and of the facts $R(p_j, 1, \phi_i, \phi_{i+1})$ such that p_j occurs positively in ϕ_i where the addition $i + 1$ is meant modulo the number m of clauses in Φ. We want to show that $\emptyset$ is a repair of r_Φ with respect to IC if and only if Φ is not satisfiable.

For the *only if* direction notice that if ρ is a satisfying assignment of Φ then the subset of r_Φ consisting of all the facts of the form $R(p, \rho(p), \phi_i, \phi_{i+1})$ is a repair, and obviously $\emptyset$ is not a repair then.

For the opposite direction first notice that a repair r' of r_Φ which is nonempty contains some fact of the form $R(\_, \_, \phi_i, \phi_{i+1})$. So, by inclusion $A_3 \subseteq A_4$ it must also contain some fact of the form $R(\_, \_, \phi_{i-1}, \phi_i)$. By induction we show that

(*) for every clause ϕ_j from Φ there is a fact of the form $R(\_, \_, \phi_j, \phi_{j+1})$ in r'.

Now we make use of the functional dependency $A_1 \to A_2$. If r' is a repair of r_Φ then for each variable p there are either only facts of the form $R(p, 0, \_, \_)$ in r' or only facts of the form $R(p, 1, \_, \_)$. Define the assignment $\rho(p)$ as 1 if there is some fact of the form $R(p, 1, \_, \_)$ in r' and as 0 otherwise. It follows from the construction of r_Φ that if a clause of the form $R(\_, \_, \phi_j, \phi_{j+1})$ is in r' then ρ satisfies ϕ_j. Together with (*) this completes the proof. ■

4.3.2 Key FDs and foreign key constraints

Theorem 10 *There exist a database schema and a set IC of integrity constraints, consisting of key FDs and foreign key constraints, such that B_{IC} is co-NP-hard.*

Proof. Again we consider checking whether the empty set is a repair. The schema consists of 10 binary relations: $R(A, B), R_{i,j}(A_{i,j}, B_{i,j})$ with $1 \leq i, j \leq 3$. For each pair (i, j) both

the key dependencies $A_{i,j} \to B_{i,j}$ and $B_{i,j} \to A_{i,j}$ are in IC, with $A_{i,j}$ as the primary key of the respective relation. The relation R is constrained by a single key dependency $B \to A$. The inclusion constraints are $B_{i,j} \subseteq B$, for each pair i, j and $A \subseteq A_{i,j}$, also for each pair i, j.

Consider a propositional formula $\Phi = \phi_1 \wedge \phi_2 \wedge \ldots \phi_m$, where ϕ_i are clauses. We assume that none of the clauses in Φ contains more than 3 literals, that each variable occurs at most 3 times in Φ, and that the number of variables in Φ is equal to the number m of clauses in the formula. It is easy to prove that satisfiability is NP-complete even for formulae of this kind. For the formula Φ we built a database instance r_Φ: in the the relation R we remember the formula Φ: it consists of such pairs (w, ϕ) that w is a literal, ϕ is a clause from Φ and w occurs in ϕ. The definitions of the relations $R_{i,j}$ are a little bit more complicated. The relation $R_{i,j}$ consists of $2m$ tuples $(p_l, \phi_{s(i,j,l)})$, and $(\neg p_l, \phi_{s(i,j,l)})$, with s still to be defined, will be a function from $\{1, 2, 3\} \times \{1, 2, 3\} \times \{1, 2, \ldots m\}$ to $\{1, 2, \ldots m\}$ and, more precisely, it is going to be a permutation of $\{1, 2, \ldots m\}$ for every fixed pair (i, j). Define $s(i, j, l)$ as n if p_l (or $\neg p_l$) occurs in the clause ϕ_{n+1} (where addition is modulo the number of clauses m), if p_l is the ith variable in this clause, and if it is jth occurrence of p_l in Φ. Now, for each (i, j) let $s(i, j, \_)$ be any permutation consistent with the above definition. It follows directly from our construction that:

Lemma 5 *For each clause ϕ_n from Φ and for each variable p occurring in ϕ_n there is a relation $R_{i,j}$ such that the tuples (p, ϕ_{n-1}) and $(\neg p, \phi_{n-1})$ are in $R_{i,j}$.*

We want to show that $\emptyset$ is a repair of r_Φ with respect to IC if and only if Φ is not satisfiable.

The *only if* direction is simple. Assume that Φ is satisfiable and let ρ be a satisfying assignment. In each tuple in each of the relations $R, R_{i,j}$ in r_Φ the first argument is always a literal. Let r' be a subset of r_Φ consisting of such facts $R(w, \phi)$ or $R_{i,j}(w, \phi)$ that $\rho(w) = 1$. The key constraints for $R_{i,j}$ are satisfied in r'. The inclusion constraints $B_{i,j} \subseteq B$ are satisfied because, since ρ was an assignment satisfying Φ, $B = \{\phi_1, \phi_2, \ldots \phi_m\}$. Also the inclusions $A \subseteq A_{i,j}$ hold. But the key dependency $B \to A$ does not need to hold in r' (this is because there is possibly more than one literal w in some clause such that $\rho(w) = 1$). To construct a nonempty repair of r_Φ take now r'' built with the same relations $R_{i,j}$ as r' and with relation R being the result of selecting from the relation R in r' exactly one tuple (w, ϕ) for each ϕ.

The *if* direction is more complicated. If r' is a repair of r_Φ then, in each of the relations $R_{i,j}$, for each clause $s(i, j, l)$ at most one of the tuples $(p_l, \phi_{s(i,j,l)})$ and $(\neg p_l, \phi_{s(i,j,l)})$ can be in $R_{i,j}$. This implies that at most one of the literals $p_l, \neg p_l$ can be in $A_{i,j}$. But $A \subseteq A_{i,j}$ and, since Φ is not satisfiable, there must be a clause ϕ_l such that none of the literals from ϕ_l is in A. This means that ϕ_l is not in B. Consider the clause ϕ_{l+1}. By Lemma 5 for each variable p from ϕ_{l+1} there is a relation $R_{i,j}$ such that the tuples (p, ϕ_l) and $(\neg p, \phi_l)$ are in $R_{i,j}$ in r_Φ. But, by the inclusion constraints, each of the $B_{i,j}$ should be a subset of B, so since ϕ_l is not in B in r' it is also not in any of the $B_{i,j}$ in r'. While removing ϕ_l from $B_{i,j}$ we also delete the variables occurring in a tuple of $R_{i,j}$ together with ϕ_l. This means that for each variable p from the clause ϕ_{l+1} there is a relation $R_{i,j}$ such that neither p nor $\neg p$ is in $A_{i,j}$. But A is a subset of each of the $A_{i,j}$. This means that none of the literals from ϕ_{l+1}

can be in A. So ϕ_{l+1} cannot be in B! Now, using this argument m times we can remove all the tuples from the relations, thus proving that r' is empty. ■

4.4 Arbitrary FDs and INDs

Theorem 11 *The repair checking problem for arbitrary FDs and INDs is co-NP-complete.*

Proof. Co-NP-hardness was established earlier in this section. The membership in co-NP follows from the definition of repair. ■

Theorem 12 *The consistent query answers problem for arbitrary FDs and INDs is* Π_2^p*-complete.*

Proof. The membership in Π_2^p follows from the definition of consistent query answer. We show Π_2^p-hardness below. Consider a quantified boolean formula ϕ of the form

$$\forall p_1, p_2, \ldots p_k \exists q_1, q_2, \ldots q_l \ \psi$$

where ψ is quantifier-free and equals to $\psi_1 \wedge \psi_2 \wedge \ldots \psi_m$, where ψ_i are clauses. We will construct a database instance r_ϕ, over a schema with a single relation $R(A, B, C, D)$, such that $R(a, a, \psi_1, a)$ is a consistent answer if and only if ϕ is true. The integrity constraints will be $A \rightarrow B$ and $C \subseteq D$.

There are 3 kinds of tuples in r_ϕ. For each occurence of a literal in ψ we have one tuple of the first kind (we adopt the convention that ψ_{m+1} is ψ_1):

- $R(p_i, 1, \psi_j, \psi_{j+1})$ if p_i occurs positively in ψ_j,
- $R(q_i, 1, \psi_j, \psi_{j+1})$ if q_i occurs positively in ψ_j,
- $R(p_i, 0, \psi_j, \psi_{j+1})$ if p_i occurs negatively in ψ_j,
- $R(q_i, 0, \psi_j, \psi_{j+1})$ if q_i occurs negatively in ψ_j.

For each universally quantified variable p_i we have two tuples of the second kind: $R(p_i, 1, a_i, a_i)$ and $R(p_i, 0, a_i, a_i)$. Finally, there is just one tuple of the third kind: $R(a, a, \psi_1, a)$.

Let us first show that if ϕ is false then $R(a, a, \psi_1, a)$ is not a consistent answer. Let σ be such a valuation of the variables $p_1, p_2, \ldots p_k$ that the formula $\sigma(\phi)$ (with free variables $q_1, q_2, \ldots q_l$ is not satisfiable. It will be enough to show that the set s_σ of all the tuples from r_ϕ which are of the form $R(p_1, \sigma(p_i), a_i, a_i)$ is a repair. The set s_σ is consistent. So if it is not a repair then another consistent subset $s \supset s_\sigma$ of r_ϕ must exist. Due to the FD s does not contain any tuple of the second kind not being already in s_σ. So, there must be some tuple of the first or the third kind in s. But that means (due to the IND) that for each ψ_j there is either some tuple of the form $R(p_i, \sigma(p_i), \psi_j, \psi_{j+1})$ in s , or some tuple of the form $R(p_i, \varepsilon_i, \psi_j, \psi_{j+1})$, where $\varepsilon_i \in \{0, 1\}$. Due to the FD, for each q_i there can be at most one such ε_i. Define $\bar{\sigma}(q_i) = \varepsilon_i$. Then $\bar{\sigma}(\sigma(\phi)) = 1$ which is impossible.

For the opposite direction suppose that ϕ is true but $R(a, a, \psi_1, a)$ is not a consistent answer. The last means that there exists a repair s of r_ϕ such that no tuple of the form

$R(\_, \_, \psi_1, \_)$ can be found in s. But this implies that there are no tuples of the first kind in s, and so s only consists of some tuples of the second kind. Due to the FD there exists a valuation σ such that s consists of all the tuples of the second kind which are of the form $R(p_1, \sigma(p_i), a_i, a_i)$. Since ϕ is true, there exists a valuation $\bar{\sigma}$ of variables $q_1, q_2, \ldots q_l$ such that $\bar{\sigma}(\sigma(\phi)) = 1$. But then the set s' consisting of all the tuples from s, $R(a, a, \psi_1, a)$, and all the tuples of the first kind which are either of the form $R(p_i, \sigma(p_i), \psi_j, \psi_{j+1})$ or $R(q_i, \bar{\sigma}(q_i), \psi_j, \psi_{j+1})$ is consistent, which contradicts the assumption that s is a repair. ■

5 Related work

We only briefly survey the related work here. A more comprehensive discussion can be found in [ABC99, BC03].

There are several similarities between our approach to consistency handling and those followed by the belief revision/update community [GR95]. Database repairs (Definition 4) coincide with revised models defined by Winslett in [Win88]. The treatment in [Win88] is mainly propositional, but a preliminary extension to first order knowledge bases can be found in [CW94]. Those papers concentrate on the computation of the models of the revised theory, i.e., the repairs in our case. Comparing our framework with that of belief revision, we have an empty domain theory, one model: the database instance, and a revision by a set of ICs. The revision of a database instance by the ICs produces new database instances, the repairs of the original database. The complexity of belief revision (and the related problem of counterfactual inference which corresponds to our computation of consistent query answers) in the propositional case was exhaustively classified by Eiter and Gottlob [EG92]. Among the constraint classes considered in the current paper, only denial constraints can be represented propositionally by grounding. However, such grounding results in an unbounded update formula, which prevents the transfer of any of the PTIME upper bounds from [EG92] into our framework. Similarly, their lower bounds require different kinds of formulas from those that we use.

The need to accommodate violations of functional dependencies is one of the main motivations for considering disjunctive databases [INV91, vdM98] and has led to various proposals in the context of data integration [AKWS95, BKMS92, Dun96, LM96]. There seems to be an intriguing connection between relation repairs w.r.t. FDs and databases with disjunctive information [vdM98]. For example, the set of repairs of the relation *Person* from Example 6 can be represented as a disjunctive database D consisting of the formulas

$$\mathit{Person}(\text{Brown}, \text{Amherst}, 115\ \text{Klein}) \vee \mathit{Person}(\text{Brown}, \text{Amherst}, 120\ \text{Maple})$$

and

$$\mathit{Person}(\text{Green}, \text{Clarence}, 4000\ \text{Transit}).$$

Each repair corresponds to a minimal model of D and vice versa. We conjecture that the set of all repairs of an instance w.r.t. a set of FDs can be represented as a disjunctive table (with rows that are disjunctions of atoms with the same relation symbol). The relationship in the other direction does not hold, as shown by the folowing example [ABC$^+$03].

Example 7 *The set of minimal models of the formula*

$$(p(a_1, b_1) \vee p(a_2, b_2)) \wedge p(a_3, b_3)$$

cannot be represented as a set of repairs of any set of FDs. □

Known tractable classes of first-order queries over disjunctive databases typically involve conjunctive queries and databases with restricted OR-objects [INV91, IvdMV95]. In some cases, like in Example 6, the set of all repairs can be represented as a table with OR-objects. But in general this is not the case [ABC$^+$03].

Example 8 *Consider the following set of FDs $F = \{A \to B, A \to C\}$, which is in BCNF. The set of all repairs of the instance $\{(a_1, b_1, c_1), (a_1, b_2, c_2)\}$ cannot be represented as a table with OR-objects.* □

The relationship in the other direction, from tables with OR-objects to sets of repairs, also does not hold.

Example 9 *Consider the following table with OR-objects:*

OR(a,b)	*c*
a	*OR(c,d)*

It does not represent the set of all repairs of any instance under any set of FDs. □

In general, a correspondence between sets of repairs and tables with OR-objects holds only in the very restricted case when the relation is binary, say $R(A, B)$, and there is one FD $A \to B$. The paper [IvdMV95] contains a complete classification of the complexity of conjunctive queries for tables with OR-objects. It is shown how the complexity depends on whether the tables satisfy various schema-level criteria, governing the allowed occurrences of OR-objects. Since there is no exact correspondence between tables with OR-objects and sets of repairs of a given database instance, the results of [IvdMV95] do not directly translate to our framework, and vice versa.

There are several proposals for language constructs specifying nondeterministic queries that are related to our approach (*witness* [AHV95], *choice* [GGSZ97, GP98, GSZ95]). Essentially, the idea is to construct a maximal subset of a given relation that satisfies a given set of functional dependencies. Since there is usually more than one such subset, the approach yields nondeterministic queries in a natural way. Clearly, maximal consistent subsets (choice models [GGSZ97]) correspond to repairs. Datalog with choice [GGSZ97] is, in a sense, more general than our approach, since it combines enforcing functional dependencies with inference using Datalog rules. Answering queries in all choice models ($\forall G$-queries [GSZ95]) corresponds to our notion of computation of consistent query answers (Definition 6). However, the former problem is shown to be co-NP-complete and no tractable cases are identified. One of the sources of complexity in this case is the presence of Datalog rules, absent from our approach. Moreover, the procedure proposed in [GSZ95] runs in exponential time if there are exponentially many repairs, as in Example 6. Also, only conjunctions of literals are considered as queries in [GSZ95].

A purely proof-theoretic notion of consistent query answer comes from Bry [Bry97]. This notion, described only in the propositional case, corresponds to evaluating queries after all the tuples involved in inconsistencies have been eliminated. The paper [ABC99] introduced the notions of repair and consistent query answer used in the current research. It proposed computing consistent query answers through *query transformation*. The papers

[ABC01, ABC$^+$03] studied the computation of consistent query answers in the context of FDs and scalar aggregation queries.

Wijsen [Wij03] studied the problem of consistent query answering in the context of universal constraints. In contrast to Definition 4, he considers repairs obtained by modifying individual tuple components. Notice that a modification of a tuple component cannot be necessarily simulated as a deletion followed by an insertion, because this might not be minimal under set inclusion. Wijsen proposes to represent all the repairs of an instance using a single *trustable tableau*. From this tableau, answers to conjunctive queries can be efficiently obtained. It is not clear, however, what is the computational complexity of constructing the tableau, or even whether the tableau is always of polynomial size.

Representing repairs as stable models of logic programs with disjunction and classical negation has been proposed in [ABC00, GGZ01]. Those papers consider computing consistent answers to first-order queries. While the approach is very general, no tractable cases beyond those already implicit in the results of [ABC99] are identified. The semantics of referential integrity actions are captured using stable models of logic programs with negation in [LML97].

It is interesting to contrast our results in Section 4 with the classical results about the implication problem for FDs and INDs [AHV95]. This problem is undecidable in general but becomes decidable under suitable restrictions on INDs. For instance, it is decidable in PTIME if the INDs are unary and in EXPTIME if the INDs are acyclic. The problems discussed in our paper are all in Π_2^p (Section 4). The role the syntactic restrictions play in this context is different. The restriction to unary INDs is not helpful, c.f., Theorem 11. The restriction to acyclic INDs makes the repair checking problem tractable (Theorem 7) but not so the problem of consistent query answers (Theorem 8).

In [MR92], several classes of FDs and INDs were identified for which the implication problem does not exhibit any interaction between the FDs and the INDs. I.e., a set of constraints implies an FD (resp. an IND) iff the FDs (resp. the INDs) in this set imply it. Unfortunately, the syntactic restrictions on constraints that guarantee no interaction in the above sense do not play a similar role in our context. It seems that the notion of maximality present in the repair definition forces a relationship between the FDs and the INDs that is much tighter than the one implicit in the implication problem.

In [MM90, MR92], it is investigated what kind of relational schemas and integrity constraints can result from mapping an Entity-Relationship schema (this is a common way of designing relational schemas). Acyclicity of INDs is a necessary requirement, thus repair checking is tractable in this case. However, it turns out that the schema from Theorem 8 could result from such a mapping. Thus, even restricting the relational schemas to those that correspond to Entity-Relationship schemas does not guarantee the tractability of consistent query answers.

6 Conclusions and future work

In this paper we have investigated the computational complexity issues involved in minimal-change integrity maintenance using tuple deletions, in the presence of denial constraints and inclusion dependencies. We have identified several tractable cases and shown that generalizing them leads to intractability.

We envision several possible directions for future work. First, one can consider various *preference orderings* on repairs. Such orderings are often natural and may lead to further tractable cases. Some preliminary work in this direction is reported in [GGZ01]. Second, a natural scenario for applying the results developed in this paper is *query rewriting* in the presence of distributed data sources [DGL00, Hal01, Len02]. Recent work in this area has started to address the issues involved in data sources being inconsistent [BCCG02, LLR02]. Finally, as XML is playing an increased role in data integration [PV99, LPV00, DHW01], it would be interesting and challenging to develop the appropriate notions of repair and consistent query answer in the context of XML databases. Recent integrity constraint proposals for XML include [BDF+01, FS00, FKS01].

References

[ABC99] M. Arenas, L. Bertossi, and J. Chomicki. Consistent Query Answers in Inconsistent Databases. In *ACM Symposium on Principles of Database Systems*, pages 68–79, 1999.

[ABC00] M. Arenas, L. Bertossi, and J. Chomicki. Specifying and Querying Database Repairs Using Logic Programs with Exceptions. In *International Conference on Flexible Query Answering Systems*, pages 27–41. Springer-Verlag, 2000.

[ABC01] M. Arenas, L. Bertossi, and J. Chomicki. Scalar Aggregation in FD-Inconsistent Databases. In *International Conference on Database Theory*, pages 39–53. Springer-Verlag, LNCS 1973, 2001.

[ABC+03] M. Arenas, L. Bertossi, J. Chomicki, X. He, V. Raghavan, and J. Spinrad. Scalar Aggregation in Inconsistent Databases. *Theoretical Computer Science*, 2003. Special issue: selected papers from ICDT 2001, to appear.

[AHV95] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[AKWS95] S. Agarwal, A. M. Keller, G. Wiederhold, and K. Saraswat. Flexible Relation: An Approach for Integrating Data from Multiple, Possibly Inconsistent Databases. In *IEEE International Conference on Data Engineering*, 1995.

[BC03] L. Bertossi and J. Chomicki. Query Answering in Inconsistent Databases. In J. Chomicki, R. van der Meyden, and G. Saake, editors, *Logics for Emerging Applications of Databases*. Springer-Verlag, 2003. To appear.

[BCCG02] L. Bertossi, J. Chomicki, A. Cortes, and C. Gutierrez. Consistent Answers from Integrated Data Sources. In *International Conference on Flexible Query Answering Systems*, Copenhagen, Denmark, October 2002. Springer-Verlag.

[BDF+01] P. Buneman, S. Davidson, W. Fan, C. Hara, and W. Tan. Keys for XML. In *International World Wide Web Conference*, 2001. Full version to appear in Computer Networks.

[BKMS92] C. Baral, S. Kraus, J. Minker, and V. S. Subrahmanian. Combining Knowledge Bases Consisting of First-Order Theories. *Computational Intelligence*, 8:45–71, 1992.

[Bry97] F. Bry. Query Answering in Information Systems with Integrity Constraints. In *IFIP WG 11.5 Working Conference on Integrity and Control in Information Systems*. Chapman &Hall, 1997.

[CH80] A. K. Chandra and D. Harel. Computable Queries for Relational Databases. *Journal of Computer and System Sciences*, 21:156–178, 1980.

[CM77] A. Chandra and P. Merlin. Optimal Implementation of Conjunctive Queries in Relational Databases. In *ACM SIGACT Symposium on the Theory of Computing*, pages 77–90, 1977.

[CM02] J. Chomicki and J. Marcinkowski. On the Computational Complexity of Consistent Query Answers. Technical Report arXiv:cs.DB/0204010, arXiv.org e-Print archive, April 2002.

[CW94] T. Chou and M. Winslett. A Model-Based Belief Revision System. *Journal of Automated Reasoning*, 12:157–208, 1994.

[Dat81] C. J. Date. Referential Integrity. In *International Conference on Very Large Data Bases*, pages 2–12, 1981.

[DGL00] O.M. Duschka, M.R. Genesereth, and A.Y. Levy. Recursive Query Plans for Data Integration. *Journal of Logic Programming*, 43(1):49–73, 2000.

[DHW01] D. Draper, A. Halevy, and D. Weld. The Nimble XML Data Integration System. In *ACM SIGMOD International Conference on Management of Data*, 2001.

[Dun96] Phan Minh Dung. Integrating Data from Possibly Inconsistent Databases. In *International Conference on Cooperative Information Systems*, Brussels, Belgium, 1996.

[EG92] T. Eiter and G. Gottlob. On the Complexity of Propositional Knowledge Base Revision, Updates, and Counterfactuals. *Artificial Intelligence*, 57(2-3):227–270, 1992.

[FKS01] W. Fan, G. Kuper, and J. Simeon. A Unified Constraint Model for XML. In *International World Wide Web Conference*, 2001. Full version to appear on Computer Networks.

[FS00] W. Fan and J. Simeon. Integrity Constraints for XML. In *ACM Symposium on Principles of Database Systems*, 2000. Full version to appear in JCSS.

[GGSZ97] F. Giannotti, S. Greco, D. Sacca, and C. Zaniolo. Programming with Nondeterminism in Deductive Databases. *Annals of Mathematics and Artificial Intelligence*, 19(3-4), 1997.

[GGZ01] G. Greco, S. Greco, and E. Zumpano. A Logic Programming Approach to the Integration, Repairing and Querying of Inconsistent Databases. In *International Conference on Logic Programming*, pages 348–364. Springer-Verlag, LNCS 2237, 2001.

[GP98] F. Giannotti and D. Pedreschi. Datalog with Non-deterministic Choice Computes NDB-PTIME. *Journal of Logic Programming*, 35:75–101, 1998.

[GR95] P. Gärdenfors and H. Rott. Belief Revision. In D. M. Gabbay, J. Hogger, C, and J. A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 4, pages 35–132. Oxford University Press, 1995.

[GSZ95] S. Greco, D. Sacca, and C. Zaniolo. Datalog Queries with Stratified Negation and Choice: from P to D^P. In *International Conference on Database Theory*, pages 82–96. Springer-Verlag, 1995.

[Hal01] A. Y. Halevy. Answering Queries Using Views: A Survey. *VLDB Journal*, 10(4):270–294, 2001.

[INV91] T. Imieliński, S. Naqvi, and K. Vadaparty. Incomplete Objects - A Data Model for Design and Planning Applications. In *ACM SIGMOD International Conference on Management of Data*, pages 288–297, Denver, Colorado, May 1991.

[IvdMV95] T. Imieliński, R. van der Meyden, and K. Vadaparty. Complexity Tailored Design: A New Design Methodology for Databases With Incomplete Information. *Journal of Computer and System Sciences*, 51(3):405–432, 1995.

[Len02] M. Lenzerini. Data Integration: A Theoretical Perspective. In *ACM Symposium on Principles of Database Systems*, 2002. Invited talk.

[LLR02] D. Lembo, M. Lenzerini, and R. Rosati. Source Inconsistency and Incompleteness in Data Integration. In *Workshop on Nonmonotonic Reasoning (NMR'02)*, Toulouse, France, 2002.

[LM96] J. Lin and A. O. Mendelzon. Merging Databases under Constraints. *International Journal of Cooperative Information Systems*, 7(1):55–76, 1996.

[LML97] B. Ludäscher, W. May, and G. Lausen. Referential Actions as Logical Rules. In *ACM Symposium on Principles of Database Systems*, pages 217–227, 1997.

[LPV00] B. Ludäscher, Y. Papakonstantinou, and P. Velikhov. Navigation-Driven Evaluation of Virtual Mediated Views. In *International Conference on Extending Database Technology*, 2000.

[MM90] V. M. Markowitz and J.A. Makowsky. Identifying Extended Entity-Relationship Object Structures in Relational Schemas. *IEEE Transactions on Software Engineering*, 16(8):777–790, 1990.

[MR92] H. Mannila and K-J. Räihä. *The Design of Relational Databases*. Addison-Wesley, 1992.

[MS02] Jim Melton and Alan R. Simon. *SQL:1999 Understanding Relational Language Components*. Morgan Kaufmann, 2002.

[PV99] Y. Papakonstantinou and V. Vassalos. Rewriting Queries Using Semistructured Views. In *ACM SIGMOD International Conference on Management of Data*, 1999.

[Var82] M. Y. Vardi. The Complexity of Relational Query Languages. In *ACM Symposium on Theory of Computing*, pages 137–146, 1982.

[vdM98] R. van der Meyden. Logical Approaches to Incomplete Information: A Survey. In J. Chomicki and G. Saake, editors, *Logics for Databases and Information Systems*, chapter 10. Kluwer Academic Publishers, Boston, 1998.

[Wij03] J. Wijsen. Condensed Representation of Database Repairs for Consistent Query Answering. In *International Conference on Database Theory*, 2003.

[Win88] M. Winslett. Reasoning about Action using a Possible Models Approach. In *National Conference on Artificial Intelligence*, 1988.