Containment of Conjunctive Queries over Conceptual Schemata

Andrea Calì

Faculty of Computer Science, Free University of Bozen-Bolzano,
piazza Domenicani 3, I-39100 Bolzano, Italy
ac@andreacali.com

Abstract. Conceptual Modelling plays a fundamental role in database design since Chen's Entity-Relationship (ER) model. In this paper we consider a conceptual model capable of capturing classes of objects with their attributes, relationships among classes, cardinality constraints in the participation of entities to relationships, and is-a relations among both classes and relationships. We provide a formal semantics for such model in terms of predicates and constraints over their extensions. We address the problem of containment of conjunctive queries over a conceptual schema, and we show an algorithm for solving the problem, that achieves better computational complexity than the techniques found in the literature. The results presented here are directly applicable in query answering on incomplete databases, and in data integration under constraints.

1 Introduction

Conceptual models, and in particular the Entity-Relationship (ER) model [9], play a fundamental role in database design. Conceptual schemata used in database design have the necessary expressiveness and flexibility for effectively representing the domain of interest, and are precise enough to allow the implementation on DBMSs.

In this paper we address the problem of query containment, where queries are conjunctive queries expressed over a conceptual schema. As a conceptual model, we adopt a formalism that we call *Extended Entity-Relationship (EER) Model*, able to represent classes of objects with their attributes, relationships among classes, cardinality constraints in the participation of entities to relationships, and is-a relations among both classes and relationships. Since our conceptual model deals with classes (entities) and relations (relationships) on classes, we provide a formal semantics to our conceptual model in terms of the relational database model. In our setting, conjunctive queries are expressed by using predicates (relations) appearing in the relational representation of the conceptual schema.

The problem of determining containment of queries is highly relevant for query optimisation [8]; in general a query Q_1 is contained in another query Q_2 if for every database D the answers to Q_1 evaluated over D are a subset of the

M.L. Lee, K.L. Tan, and V. Wuwongse (Eds.): DASFAA 2006, LNCS 3882, pp. 628–643, 2006.

answers to Q_2 evaluated over D. The query containment problem is complicated, in our setting, by the high expressiveness of the EER model. In fact, we represent a conceptual schema by means of a relational schema, on whose predicates the queries are formulated, and therefore we need to make use of *integrity constraints* to capture the expressiveness of the EER model.

The problem of determining whether a query Q_1 is contained in a query Q_2 under a set Σ of constraints, written $Q_1 \subseteq_\Sigma Q_2$, consists in determining whether *for every database D satisfying Σ* the answers to Q_1 evaluated over D are a subset of the answers to Q_2 evaluated over D. Consider the following example, adapted from [11] and entirely based on the relational database model. We have two relations

$$\begin{array}{ll} \mathsf{employee} & [emp\_no, emp\_name, salary, dept] \\ \mathsf{dept} & [dept\_no, dept\_name, location] \end{array}$$

with a single integrity constraint $\mathsf{employee}[4] \subseteq \mathsf{dept}[1]$, stating that every department number appearing in the fourth column of employee must be the number of some department, therefore it must appear in the first column of dept. Now, consider the two conjunctive queries

$$Q_1(U) \leftarrow \mathsf{employee}(U, agenor, X, Y)$$
$$Q_2(U) \leftarrow \mathsf{employee}(U, agenor, X, Y), \mathsf{dept}(Y, Z, W)$$

Without constraints we have that Q_1 is not contained in Q_2, while in the presence of the constraint the queries are equivalent, i.e. they are contained in each other.

In the rest of the paper we will present an algorithm that checks containment of queries expressed over an EER schema, represented by means of a relational schema with constraints. The class of constraints we deal with does not fall in the class of IDs and FDs for which containment is known to be decidable (see [4]); indeed, the decidability of the problem is already known from a work that addresses containment in the context of a Description Logics that is able to capture the EER model [6]. However, our technique, besides providing an in-depth look at the issue of containment of queries over EER schemata, yelds an upper bound for the complexity of the problem that is better than the one of [6].

2 Preliminaries

In this section we give a formal definition of the relational data model, of database constraints, of conjunctive queries, and of containment of conjunctive queries under constraints.

The relational data model. In the relational data model [10], predicate symbols are used to denote the relations in the database, whereas constant symbols denote the objects and the values stored in relations. We assume to have two distinct, fixed and infinite alphabets Γ and Γ_f of constants and *fresh constants* respectively, and we consider only databases over $\Gamma \cup \Gamma_f$. We adopt the so-called *unique name assumption*, i.e. we assume that different constants denote different objects.

A *relational schema* $\mathcal{R}$ consists of an alphabet of *predicate* (or *relation*) symbols, each one with an associated arity denoting the number of arguments of the predicate (or attributes of the relation). When a relation symbol R has arity n, it can be denoted by R/n.

A *relational database* (or simply database) D over a schema $\mathcal{R}$ is a set of relations with constants as atomic values. We have one relation of arity n for each predicate symbol of arity n in the alphabet $\mathcal{R}$. The relation R^D in D corresponding to the predicate symbol R consists of a set of tuples of constants, that are the tuples satisfying the predicate R in D.

When, given a database D for a schema $\mathcal{R}$, a tuple $t = (c_1, \ldots, c_n)$ is in R^D, where $R \in \mathcal{R}$, we say that the fact $R(c_1, \ldots, c_n)$ holds in D. Henceforth, we will use interchangeably the notion of fact and tuple.

Integrity constraints. *Integrity constraints* are assertions on the symbols of the alphabet $\mathcal{R}$ that are intended to be satisfied in every database for the schema. The notion of satisfaction depends on the type of constraints defined over the schema. A database D over a schema $\mathcal{R}$ is said to *satisfy* a set of integrity constraints Σ expressed over $\mathcal{R}$, written $D \models \Sigma$, if every constraint in Σ is satisfied by D.

The database constraints of our interest are *functional dependencies (FDs)*, *inclusion dependencies (IDs)* and *key dependencies (KDs)* (see e.g. [2]). We denote with boldface uppercase letters (e.g. $\mathbf{X}$) both sequences and sets of attributes of relations. Given a tuple t in relation R^D, i.e. a fact $R(t)$ in a database D for a schema $\mathcal{R}$, and a set of attributes $\mathbf{X}$ of R, we denote with $t[\mathbf{X}]$ the *projection* (see e.g. [2]) of t on the attributes in $\mathbf{X}$.

(i) Functional dependencies (FDs). A functional dependency on a relation R is denoted by $R : \mathbf{X} \to \mathbf{Y}$. Such a constraint is satisfied in a database D iff for each $t_1, t_2 \in R^D$ we have that if $t_1[\mathbf{X}] = t_2[\mathbf{X}]$ then $t_1[\mathbf{Y}] = t_2[\mathbf{Y}]$.

(ii) Inclusion dependencies (IDs). An inclusion dependency between relations R_1 and R_2 is denoted by $R_1[\mathbf{X}] \subseteq R_2[\mathbf{Y}]$. Such a constraint is satisfied in a database D iff for each tuple t_1 in R_1^D there exists a tuple t_2 in R_2^D such that $t_1[\mathbf{X}] = t_2[\mathbf{Y}]$.

(iii) Key dependencies (KDs). A key constraint over relation R is denoted by $key(R) = \mathbf{K}$, where $\mathbf{K}$ is a subset of the attributes of R. Such a constraint is satisfied in a database D iff for each $t_1, t_2 \in R^D$ we have $t_1[\mathbf{K}] \neq t_2[\mathbf{K}]$. Observe that this constraints is equivalent to the functional dependency $R : \mathbf{K} \to \mathbf{A}_R$, where $\mathbf{A}_R$ is the set of all attributes of R, therefore KDs are a special case of FDs.

Queries. A *relational query* is a formula that specifies a set of data to be retrieved from a database. In the following we will refer to the class of *conjunctive queries*. A *conjunctive query* (CQ) Q of arity n over a schema $\mathcal{R}$ is written in the form $Q(\boldsymbol{X}) \leftarrow body(\boldsymbol{X}, \boldsymbol{Y})$ where:

(1) Q belongs to a new alphabet $\mathcal{Q}$ (the alphabet of queries, that is disjoint from both Γ, Γ_f and $\mathcal{R}$);
(2) $Q(\boldsymbol{X})$ is the *head* of the conjunctive query, denoted *head*(Q);
(3) *body*$(\boldsymbol{X}, \boldsymbol{Y})$ is the *body* of the conjunctive query, denoted *body*(Q), and is a conjunction of atoms involving the variables $\boldsymbol{X} = X_1, \ldots, X_n$ and $\boldsymbol{Y} = Y_1, \ldots, Y_m$, and constants from Γ;
(4) the predicate symbols of the atoms are in $\mathcal{R}$,
(5) the number of variables of $\boldsymbol{X}$ is called the *arity* of Q.

Every variable appearing more than once in Q (more than once in the body, or both in the body and in the head) is called *distinguished variable (DV)*; every othervariable is called *non-distinguished variable (NDV)*. We denote with $Var(Q)$ the set of all variables of Q.

Given a database D, the answer to Q over D, denoted $Q(D)$, is the set of n-tuples of constants $(c_1, \ldots, c_n)$, such that, when substituting each x_i with c_i, for $1 \leq i \leq n$, the formula $\exists \boldsymbol{Y} . body(\boldsymbol{X}, \boldsymbol{Y})$ evaluates to *true* in D, where $\exists \boldsymbol{Y}$ is a shorhand for $\exists Y_1 \cdots \exists Y_m$.

Query containment. Given two CQs Q_1, Q_2 over a relational schema $\mathcal{R}$, we say that Q_1 is *contained in* Q_2, denoted $Q_1 \subseteq Q_2$, if for every database D for $\mathcal{R}$ we have $Q_1(D) \subseteq Q_2(D)$. Given two CQs Q_1, Q_2 over a relational schema $\mathcal{R}$, and a set Σ of constraints on $\mathcal{R}$, we say that Q_1 is *contained in* Q_2 *under* Σ, denoted $Q_1 \subseteq_\Sigma Q_2$, if for every database D for $\mathcal{R}$ we have that $D \models \Sigma$ implies $Q_1(D) \subseteq Q_2(D)$.

3 The Conceptual Model

In this section we present the conceptual model we shall deal with in the rest of the paper, and we give its semantics in terms of relational database schemata with constraints.

Our model incorporates the basic features of the ER model [9] and OO models, including subset (or is-a) constraints on both entities and relationships. It is an extension of the one presented in [3], and here we use a notation analogous to that of [3]. Henceforth, we will call our model *Extended Entity-Relationship (EER) model*, and we will call schemata expressed in the EER model *Extended Entity-Relationship (EER) schemata*.

An *EER schema* consists of a collection of entity, relationship, and attribute definitions over an *alphabet Sym of symbols*. The alphabet *Sym* is partitioned into a set of entity symbols (denoted by *Ent*), a set of relationship symbols (denoted by *Rel*), and a set of attribute symbols (denoted by *Att*).

An *entity definition* has the form

entity E
  isa: $E_1, \ldots, E_h$
  participates(≥ 1): $R_1 : c_1, \ldots, R_\ell : c_\ell$
  participates(≤ 1): $R'_1 : c'_1, \ldots, R'_{\ell'} : c'_{\ell'}$

where: *(i)* $E \in \mathit{Ent}$ is the entity to be defined; *(ii)* the isa clause specifies a set of entities to which E is related via is-a (i.e., the set of entities that are supersets of E); *(iii)* the $\mathsf{participates}(\geq 1)$ clause specifies those relationships to which an instance of E must necessarily participate; and for each relationship R_i, the clause specifies that E participates as c_i-th component to R_i; *(iv)* the $\mathsf{participates}(\leq 1)$ clause specifies those relationships to which an instance of E cannot participate more than once (components are specified as in the previous case). The isa, $\mathsf{participates}(\geq 1)$ and $\mathsf{participates}(\leq 1)$ clauses are optional. A *relationship definition* has the form

$$\begin{array}{l} \mathsf{relationship}\ R\ \mathsf{among}\ E_1, \ldots, E_n \\ \quad \mathsf{isa}\colon R_1[j_{1\,1}, \ldots, j_{1\,n}], \ldots, R_h[j_{h\,1}, \ldots, j_{h\,n}] \end{array}$$

where: *(i)* $R \in \mathit{Rel}$ is the relationship to be defined; *(ii)* the entities of Ent listed in the among clause are those among which the relationship is defined (i.e., component i of R is an instance of entity E_i); *(iii)* the isa clause specifies a set of relationships to which R is related via is-a; for each relation R_i, we specify in square brackets how the components $[1, \ldots, n]$ are related to those of E_i, by specifying a permutation $[j_{i\,1}, \ldots, j_{i\,n}]$ of the components of E_i; *(iv)* the number n of entities in the among clause is the *arity* of R. The isa, clause is optional. An *attribute definition* has the form

$$\begin{array}{l} \mathsf{attribute}\ A\ \mathsf{of}\ X \\ \quad \mathit{qualification} \end{array}$$

where: *(i)* $A \in \mathit{Att}$ is the attribute to be defined; *(ii)* X is the entity or relationship to which the attribute is associated; *(iii) qualification* consists of none, one, or both of the keywords $\mathsf{functional}$ and $\mathsf{mandatory}$, specifying respectively that each instance of X has a unique value for attribute A, and that each instance of X needs to have at least a value for attribute A. If the $\mathsf{functional}$ or $\mathsf{mandatory}$ keywords are missing, the attribute is assumed by default to be multivalued and optional, respectively.

For the sake of simplicity, and without any loss of generality, we assume that in our EER model different entities and relationships have disjoint sets of attributes; also, we do not consider the domains of the attributes, i.e. the specification of the domains to which values of attributes must belong.

The semantics of an EER schema is defined by specifying when a database for that schema satisfies all constraints imposed by the constructs of the schema. First of all, we formally define a database schema from an EER diagram. Such a database schema is defined in terms of *predicates*, that represent the so-called concepts (entities, relationships and attributes) of the conceptual schema. Therefore, we define a relational database schema that encodes the properties of the EER schema $\mathcal{C}$.

(a) Each entity E in $\mathcal{C}$ has an associated predicate E of arity 1. Informally, a fact of the form $E(c)$ asserts that c is an instance of entity E.
(b) Each attribute A for an entity E in $\mathcal{C}$ has an associated predicate A of arity 2. Informally, a fact of the form $A(c, d)$ asserts that d is the value of attribute A associated to c, where c is an instance of entity E.

(c) Each relationship R among the entities $E_1, \ldots, E_n$ in $\mathcal{C}$ has an associated predicate R of arity n. Informally, a fact of the form $R(c_1, \ldots, c_n)$ asserts that $(c_1, \ldots, c_n)$ is an instance of relationship R, where $c_1, \ldots, c_n$ are instances of $E_1, \ldots, E_n$ respectively.
(d) Each attribute A for a relationship R among the entities $E_1, \ldots, E_n$ in $\mathcal{C}$ has an associated predicate A of arity $n + 1$. Informally, a fact of the form $A(c_1, \ldots, c_n, d)$ asserts that $(c_1, \ldots, c_n)$ is an instance of relationship R and d is the value of attribute A associated to $(c_1, \ldots, c_n)$.

Once we have defined the database schema $\mathcal{R}$ for an EER schema $\mathcal{C}$, we give the semantics of each construct of the EER model; this is done by specifying what databases (i.e. extension of the predicates of $\mathcal{R}$) *satisfy* the constraints imposed by the constructs of the EER diagram. We do that by making use of the relational database constraints introduced in Section 2.

(1) For each attribute $A/2$ for an entity E in an attribute definition in $\mathcal{C}$, we have the ID $A[1] \subseteq E[1]$.
(2) For each attribute $A/(n+1)$ for a relationship R/n in an attribute definition in $\mathcal{C}$, we have the ID $A[1, \ldots, n] \subseteq R[a, \ldots, n]$.
(3) For each relationship R involving an entity E_i as i-th component according to the corresponding relationship definition in $\mathcal{C}$, we have the ID $R[i] \subseteq E_i[1]$.
(4) For each mandatory attribute $A/2$ of an entity E in an attribute definition in $\mathcal{C}$, we have the ID $E[1] \subseteq A[1]$.
(5) For each mandatory attribute $A/(n+1)$ of a relationship R/n in an attribute definition in $\mathcal{C}$, we have the ID $R[1, \ldots, n] \subseteq A[1, \ldots, n]$.
(6) For each functional attribute $A/2$ of an entity E in an attribute definition in $\mathcal{C}$, we have the KD $\mathit{key}(A) = \{1\}$. In fact, there cannot be more than one value for attribute A that is assigned to a single instance of E.
(7) For each functional attribute $A/(n+1)$ of a relationship R/n in an attribute definition of $\mathcal{C}$, we have the KD $\mathit{key}(A) = \{1, \ldots, n\}$. In fact, there cannot be more than one value for attribute A that is assigned to a single instance of R.
(8) For each is-a relation between entities E_1 and E_2, in an entity definition in $\mathcal{C}$, we have the ID $E_1[1] \subseteq E_2[1]$. In fact, the is-a relation specifies a set containment between entities E_1 and E_2.
(9) For each is-a relation between relationships R_1 and R_2, where components $1, \ldots, n$ of R_1 correspond to components $j_1, \ldots, j_n$, in a relationship definition in $\mathcal{C}$, we have the ID: $R_1[1, \ldots, n] \subseteq R_2[j_1, \ldots, j_n]$. In fact, the is-a relation specifies a set containment between relationships R_1 and R_2.
(10) For each mandatory participation (participation with minimum cardinality 1) as c-th component of an entity E in a relationship R, specified by a clause $\mathsf{participates}\geq 1 \colon R : c$ in an entity definition in $\mathcal{C}$, we have the ID $E[1] \subseteq R[c]$.
(11) For each participation with maximum cardinality 1 as c-th component of an entity E in a relationship R, specified by a clause $\mathsf{participates}\leq 1 \colon R : c$ in an entity definition in $\mathcal{C}$, we have the ID $\mathit{key}(R) = \{c\}$

The class of constraints we obtain, which is a subclass of key and inclusion dependencies, is a novel class of relational database dependencies, that we shall

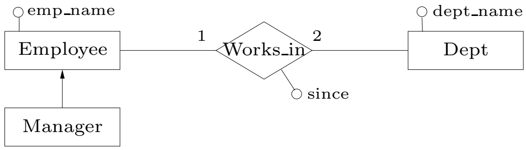

Fig. 1. EER schema for Example 1

call *conceptual dependencies (CDs)* for obvious reasons. The conjunctive queries we consider are formulated using the predicates in the relational schema we obtain from the EER schema as described above.

Example 1. Consider the EER schema shown in Figure 1, depicted in the usual graphical notation for the ER model (components are indicated for the relationship Works_in). The elements of such a schema are $\mathsf{manager}/1$, $\mathsf{employee}/1$, $\mathsf{dept}/1$, $\mathsf{works\_in}/2$, $\mathsf{emp\_name}/2$, $\mathsf{dept\_name}/2$, $\mathsf{since}/3$. The schema describes employees working in departments of a firm, and managers that are also employees. We omit the formal specification of the schema and the constraints on its relational representation. Suppose we want to know the names of the managers who work in the toy department (named *toy_dept*) since 1999. The corresponding conjunctive query is

$$\begin{aligned} Q(Z) \leftarrow\ & \mathsf{manager}(X), \mathsf{emp\_name}(X, Z), \mathsf{works\_in}(X, Y), \mathsf{since}(X, Y, 1999) \\ & \mathsf{dept}(Y), \mathsf{dept\_name}(Y, \mathit{toy\_dept}) \end{aligned}$$

4 Chase and Containment

In this section we first present the notion of *chase*, which is a fundamental tool for dealing with database constraints; then we prove some relevant properties of the chase under conceptual dependencies (CDs), by means of which we prove the decidability of the problem of containment of conjunctive queries under such dependencies.

The *chase* of a conjunctive query [13, 11] is a key concept in particular in the context of functional and inclusion dependencies. Intuitively, given a conjunctive query, its conjuncts are "frozen" and seen as facts in a database, where each variable is associated to a distinct value. Since this collection of facts in general does not satisfy the inclusion and functional dependencies, the idea is to convert the initial facts into a new set of facts constituting a database that satisfies the dependencies, possibly by collapsing facts (according to FDs) or adding new facts (according to IDs). Since a frozen query is a database, as we will see in the next section, we will define the notion of chase of a database having, in general, fresh and non-fresh constants.

Construction of the chase. Consider a database instance D for a relational schema $\mathcal{R}$, and a set Σ of dependencies on $\mathcal{R}$; in particular, $\Sigma = \Sigma_I \cup \Sigma_F$, where Σ_I is a set of inclusion dependencies and Σ_F is a set of functional dependencies.

In general, D does not satisfy Σ, written $D \not\models \Sigma$. In this case, we construct the chase of D w.r.t. Σ, denoted $chase_\Sigma(D)$, by repeatedly applying the rules defined below. We denote with $chase^*_\Sigma(D)$ the part of the chase that is already constructed before the rule is applied.

Inclusion Dependency Chase Rule. Let R, S be relational symbols in $\mathcal{R}$. Suppose there is a tuple t in $R^{chase^*_\Sigma(D)}$, and there is an ID $\sigma \in \Sigma_I$ of the form $R[\mathbf{Y}_R] \subseteq S[\mathbf{Y}_S]$. If there is no tuple t' in S^D such that $t'[\mathbf{X}_S] = t[\mathbf{X}_R]$ (in this case we say the rule is *applicable*), then we add a new tuple t_{chase} in S^D such that $t_{chase}[\mathbf{X}_S] = t[\mathbf{X}_R]$, and for every attribute A_i of S, with $1 \leq i \leq m$ and $A_i \notin \mathbf{X}_S$, $t_{chase}[A_i]$ is a fresh value in Γ_f that *follows*, according to lexicographic order, all the values already present in the chase.

Functional Dependency Chase Rule. Let R be a relational symbol in $\mathcal{R}$. Suppose there is a FD φ of the form $R: \ \mathbf{X} \rightarrow \mathbf{Y}$. If there are two tuples $t, t' \in \mathcal{R}^{chase^*_\Sigma(D)}$ such that $t[\mathbf{X}] = t'[\mathbf{X}]$ and $t[\mathbf{Y}] \neq t'[\mathbf{Y}]$ (in this case we say the rule is *applicable*), make the symbols in $t[\mathbf{Y}]$ and $t'[\mathbf{Y}]$ equal in the following way. Let $\mathbf{Y} = Y_1, \ldots, Y_\ell$; for all $i \in \{1, \ldots, \ell\}$, make $t[Y_i]$ and $t'[Y_i]$ merge into a combined symbol according to the following criterion: *(i)* if both are constants in Γ, halt the process, since the initial database cannot be chased; *(ii)* if one is in Γ and the other is a fresh constant in Γ_f, let the combined symbol be the non-fresh constant; *(iii)* if both are fresh constants in Γ_f, let the combined symbol be the one preceding the other in lexicographic order. Finally, replace all occurrences in $chase^*_\Sigma(D)$ of $t[Y_i]$ and $t'[Y_i]$ with their combined symbol.

In the following, we will need the notion of *level* of a tuple in the chase; intuitively, the lower the level of a tuple, the earlier the tuple has been constructed in the chase.

Definition 1. *Given a database instance D for a relational schema $\mathcal{R}$, and a set Σ of FDs and IDs, the* level *of a tuple t in $chase_\Sigma(D)$, denoted by $level(t)$, is defined as follows:*

(1) if t is in D then $level(t) = 0$;
(2) if t_2 is generated from t_1 by application of the ID chase rule, and $level(t_1) = k$, then $level(t_2) = k + 1$;
(3) if a FD is applied on a pair of tuples t_1, t_2, they keep their level, except when they are turned into the same tuple; in such a case, the new tuple gets the minimum of the levels of t_1 and t_2.

Now we come to the formal definition of the chase.

Definition 2. *We call* chase *of a relational database D for a schema $\mathcal{R}$, according to a set Σ of FDs and IDs, denoted $chase_\Sigma(D)$, the database constructed from the initial database D, by repeatedly executing the following steps, while the FD and ID chase rules are applicable.*

(1) while there are pairs of tuples on which the FD chase rule is applicable, apply the FD chase rule on a pair, arbitrarily chosen;

(2) if there are tuples on which the ID chase rule is applicable, choose the one *at the lowest level and apply the ID chase rule on it.*

As we pointed out before, the aim of the construction of the chase is to make the initial database satisfy the FDs and the IDs. This is formally stated by the following result.

Theorem 1. *Given a database schema $\mathcal{R}$ with a set Σ of FDs and IDs, and given a database D for $\mathcal{R}$, the database $chase_{\Sigma}(D)$ satisfies Σ.*

Proof. We prove the result by contradiction. We start from IDs; suppose a fact $R(c_1, \ldots, c_m)$ in $chase_{\Sigma_I}(D)$ violates an ID of the form $R[\mathbf{X}_R] \subseteq S[\mathbf{X}_S]$. This means that there is a tuple $t_R = (c_1, \ldots, c_m)$ in $R^{chase_{\Sigma}(D)}$ and there is no tuple t_S in $S^{chase_{\Sigma_I}(D)}$ such that $t_R[\mathbf{X}_R] = t_S[\mathbf{X}_S]$. But this is a contradiction, since these are exactly the conditions for the application of the chase rule for IDs, that has already been applied during the construction of $chase_{\Sigma_I}(D)$. As for FDs, suppose that two tuples t, t' in $chase_{\Sigma}(D)$ violate a FD of the form $R: \ \mathbf{X} \rightarrow \mathbf{Y}$, i.e. $t[\mathbf{X}] = t'[\mathbf{X}]$ and $t[\mathbf{Y}] \neq t'[\mathbf{Y}]$; this is the condition of application of the FD chase rule, therefore we have a contradiction, since the FD chase rule must have already been applied during the construction of the chase. This proves the claim. □

We remind the reader of the following definition (see e.g. [2]): a set Σ_I of IDs is *cyclic* if in Σ_I there is a sequence of dependencies $R_i[\mathbf{X}_i] \subseteq S_i[\mathbf{Y}_i]$, with $1 \leq i \leq n$, where $R_{i+1} = S_i$ for $1 \leq i \leq n$, and $R_1 = S_n$. Otherwise, Σ_I is said to be *acyclic*. It is easy to see that $chase_{\Sigma}(D)$ can be infinite only if the set if IDs in Σ is cyclic.

Associated to the chase, we have a *chase graph* that encodes the process of construction of the chase itself.

Definition 3. *Given a database D, and a set of inclusion dependencies Σ_I, let $chase_{\Sigma_I}(D)$ be the (possibly infinite) chase of D according to Σ_I. The* chase graph *associated to $chase_{\Sigma_I}(D)$ is a graph defined as follows.*

(i) The set of the nodes is the set of facts in $chase_{\Sigma_I}(D)$.
(ii) The edges are labelled with IDs in Σ_I.
(iii) Given two facts f_1, f_2 of $chase_{\Sigma_I}(D)$, the arc (f_1, f_2) is in the graph if f_2 is added to the chase in an application of the chase rule for a dependency $\sigma \in \Sigma_I$; in this case, the arc (f_1, f_2) is labelled by σ.
(iv) If there is a fact $f_1 = R(c_1, \ldots, c_n)$ and an ID of the form $R[\mathbf{Y}_R] \subseteq S[\mathbf{Y}_S]$, but the required fact f_2 is already in the chase, then there is a special arc from f_1 to f_2, that we will call cross-arc *according to the notation of [11].*

Notice that every chase graph, if we exclude the cross-arcs, is a forest of trees whose roots are the facts in the original database D.

Example 2. Consider the relations R and S, both of arity 2, and a set of IDs $\Sigma = \{\sigma_1, \sigma_2, \sigma_3\}$, with:

$$\sigma_1 : R[1] \subseteq S[1]$$
$$\sigma_2 : S[2] \subseteq R[1]$$
$$\sigma_3 : S[2] \subseteq S[1]$$

Let D be a database containing the facts $R(a, b)$ and $R(a, c)$. The chase graph associated to $chase_\Sigma(D)$ is shown in Figure 2, where the newly introduced values are α_i ($i = 1, 2, \ldots$) and the dashed arcs are cross-arcs.

The chase is a powerful tool for reasoning about dependencies [13, 14, 16, 11]. In the next following we will show how the chase can be used in testing the containment of queries under database dependencies.

Testing query containment with the chase. In their milestone paper about query containment under functional and inclusion dependencies [11], Johnson and Klug proved that, under FDs and IDs, a containment $Q_1 \subseteq_\Sigma Q_2$ between two conjunctive queries can be tested by verifying whether there is a query homomorphism from Q_2 to the chase of the database obtained by "freezing" Q_2, i.e. turning its conjuncts into facts. A homomorphism from a conjunctive query Q to a database D is a function f from the variables and constants appearing in a query Q to $\Gamma \cup \Gamma_f$ such that every conjunct $R(X_1, \ldots, X_n)$ (where every X_i is a variable or constant) is mapped to a fact of the form $R(c_1, \ldots, c_n)$ in D, where $c_i = f(X_i)$ for all $i \in \{1, \ldots, n\}$.

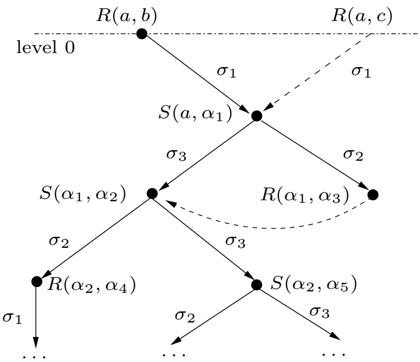

Fig. 2. Chase graph for Example 2

Definition 4. *Consider a a conjunctive query Q; the* frozen query Q*, denoted $fr(Q)$, is a pair $\langle fr(head(Q)), fr(bodyQ)\rangle$, where $\langle fr(head(Q))$ is a fact and $fr(bodyQ)\rangle$ is a database, that is obtained by choosing a homomorphism $\mu : \Gamma \cup Var(Q) \to \Gamma \cup \Gamma_f$ such that μ sends each constant of Γ into itself, and each variable in $Var(Q)$ to a fresh constant in Γ_f. Each conjunct in $body(Q)$ is sent by μ to a fact in $fr(body(Q))$, and $head(Q)$ to $fr(head(Q))$. For technical reasons, the fresh constants to which μ maps the DVs must precede in lexicographic order* all *the (fresh) constants to which μ maps the NDVs.*

Theorem 2 (see [11]). *Let Q_1, Q_2 be conjunctive queries, and Σ a set of FDs and IDs. Then $Q_1 \subseteq_\Sigma Q_2$ if and only if there is a homomorphism that sends each constant of Γ to itself, and maps $body(Q_2)$ to $chase_\Sigma(fr(body(Q_1)))$ and $head(Q_2)$ to $fr(head(Q_1))$.*

To test containment of conjunctive queries under IDs alone or *key-based* dependencies (a special class of FDs and IDs that is more general than the combination of key and foreign key dependencies), Johnson and Klug proved that it is sufficient to consider a *finite* portion of the chase; this leads to the decidability of the problem of containment, and it is also shown that the complexity of the problem of testing containment is PSPACE-complete. This result was extended in [4] to a broader class of dependencies, namely key dependencies and *non-key-conflicting*

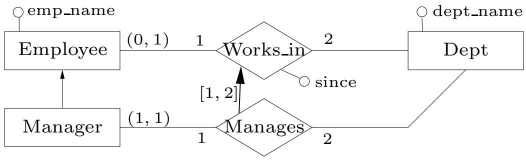

Fig. 3. EER schema for Example 3

inclusion dependencies (NKCIDs), in the context of query answering on incomplete and inconsistent databases; the NKCIDs, in fact, behave like IDs alone because they do not interfere with KDs in the construction of the chase. In our case we are in the presence of CDs, i.e. a special class of key dependencies and inclusion dependencies; IDs are not non-key-conflicting (or better *key-conflicting*), therefore the decidability of query containment is yet to be proved. In the presence of CDs, the construction of the chase presents some problems, as shown in the following example.

Example 3. Consider the EER schema shown in Figure 3, derived from that of Example 1, and depicted in the usual graphical notation for the ER model, where the label $[1, 2]$ in the is-a relation between the two relationships denotes that the components of Manages correspond, in their order, to components $1, 2$ (in this order) of Works_in, and the cardinality constraint $(0, 1)$ for Employee denotes that each instance of Employee must participate a minimum of 0 times and a mazimum of 1 times to Works_in; the cardinality constraint for the participation of Manager to Manages is analogous. We have an additional predicate $\mathsf{manages}/2$ with respect to Example 1. Suppose we have a database, obtained by freezing a query, with the facts $\mathsf{manager}(m)$ and $\mathsf{works\_in}(m, d)$. If we construct the chase, we obtain the facts $\mathsf{employee}(m)$, $\mathsf{manages}(m, \alpha_1)$, $\mathsf{works\_in}(m, \alpha_1)$, $\mathsf{dept}(\alpha_1)$, where α_1 is a fresh constant. Observe that m cannot participate more than once to $\mathsf{works\_in}$, so we deduce $\alpha_1 = d$. We must therefore replace α_1 with d in the rest of the chase, including the part that has been constructed so far.

Fortunately, also in the case of CDs, a finite portion of the chase is sufficient to test conjunctive query containment. This result can be proved analogously to the corresponding result in [11], but now things are complicated by the fact that an application of the FD chase rule can lead to a sequence of cascading applications of the same rule. This affects lower levels of the chase, so that we cannot be sure, once we stop at a certain level, whether the collapse of a pair of facts (due to the application of the FD chase rule) in a level that is far larger than the limit level can affect the portion of the chase we have constructed. In other words, in principle we do not know what the first portion of the chase actually is, before we construct the rest of the possibly infinite chase, since the application of the FD chase rule in a far level could propagate, like a crack in a high wall, down to the first portion.

First, we show that, after a certain number of levels, it is impossible that the construction of the chase of a frozen query Q_1 w.r.t. a set of CDs fails (see the

FD chase rule), leading to the conclusion that Q_1 is contained in all queries. We state our result for the chase of a database.

Lemma 1. *Let D be a database and Σ a set of CDs. We have that if the construction of $chase_\Sigma(D)$ does not fail after level $W!$, where W is the maximum width of an ID in Σ (i.e. the maximum number of attributes involved in an ID), then it does not fail in any level greater than $W!$.*

Proof (sketch). It is easy to see that the construction of the chase may fail only if the FD chase rule is applied between two tuples, containing *only* non-fresh constants in Γ, and belonging to a relation that represent a relationship of an EER schema; in fact, all other tuples of the same kind contain at most one non-fresh constant. Since tuples containing only constants in Γ propagate only through IDs that represent is-a relations between two relationships, such tuples do not "survive" after $W!$ levels. □

Lemma 2. *Let Q_1, Q_2 be two conjunctive queries, $\Sigma = \Sigma_K \cup \Sigma_I$ a set of CDs, where Σ_K and Σ_I are sets of KDs and IDs respectively. If there exists a homomorphism μ sending each constant of Γ to itself, and mapping $body(Q_2)$ to facts of $chase_\Sigma(fr(body(Q_1)))$ and $head(Q_2)$ to $fr(head(Q_1))$, then there exists another homomorphism μ' having the same properties, that sends all the facts of $body(Q_2)$ to facts in $chase_\Sigma(fr(body(Q_1)))$ appearing at levels that are lower than $|Q_2| \cdot |\Sigma| \cdot W!$, where $|Q_2|$ is the number of conjuncts in Q_2, $|\Sigma|$ is the number of dependencies in Σ, and W is the maximum width of an ID in Σ.*

Proof (sketch). The proof of this theorem goes very much like the proof of the analogous result for the case of IDs alone or *key-based* dependencies [11]: all the results hold also in the presence of CDs. We do not provide the details here, due to the fact that the proof is long and rather complicated. The only difference between our result and the result of [11] is the term $W!$, that is replaced by $(W+1)^W$ in the result of that paper. This is because $W!$ is the maximum length of a path in the chase graph made up of ordinary arcs only, starting from a fact θ, and such that there are no two *equivalent* facts in the path, where two facts θ_1, θ_2 are said to be equivalent if: *(i)* they have the incoming arc labelled with the same ID; *(ii)* for every attribute A_i, if either $\theta_1[A_i]$ or $\theta_2[A_i]$ appears in θ, it holds $\theta_1[A_i] = \theta_2[A_i]$. In our case, differently from [11], the maximum length of such a path is $W!$. □

We now come to the decidability of query containment. Our plan of attack consists in showing a *principle of locality* of KDs in the chase: in practice, we show that collapses due to the application of the FD chase rule propagate their effects at most δ levels back in the chase, where δ is a value depending on the dependencies. Therefore, in order to test whether $Q_1 \subseteq_\Sigma Q_2$, we need to construct the chase until level $\ell_{JK} = |Q_2| \cdot |\Sigma| \cdot W!$, and continue for extra δ levels; after that point, no changes will occur in the first ℓ_{JK} levels of the chase. We first need an auxiliary lemma.

Lemma 3. *Let D be a database instance (over Γ and Γ_f) for a relational schema $\mathcal{R}$, and $\Sigma = \Sigma_K \cup \Sigma_I$ a set of CDs, where Σ_K and Σ_I are sets of KDs and IDs respectively. Consider a fact θ containing a symbol $c \in (\Gamma \cup \Gamma_f)$, at level ℓ in $chase_\Sigma(D)$; then, c does not appear in any fact at levels greater than $\ell + |\Sigma| \cdot W! = \ell + \delta$.*

Proof. First, observe that only the IDs encoding is-a arcs between relationships are non-unary in Σ. Clearly, c can appear for $|\Sigma|$ more levels, but also for more, if there are cyclic non-unary IDs. If we consider a path of ordinary arcs (non-cross-arcs) corresponding to the application of the ID chase rule w.r.t. IDs $\sigma_1, \ldots, \sigma_k$ that form a cycle, if σ_i is unary for some $i \in \{1, \ldots, k\}$, c cannot survive for more than $|\Sigma|$ levels after ℓ; instead, if $\sigma_1, \ldots, \sigma_k$ are all non-unary, and therefore forced to have the same width U, the cycle of IDs formed by $\sigma_1, \ldots, \sigma_k$ can be traversed (in the application of the ID chase rule) $U!$ times, where all the generated facts are obtained by permutating the values in the U positions of θ. After that, no further propagation of c is possible. Since k is limited by $|\Sigma|$ and U by W, the thesis follows. □

Lemma 4. *Let D be a database instance (over Γ and Γ_f) for a relational schema $\mathcal{R}$, and $\Sigma = \Sigma_K \cup \Sigma_I$ a set of CDs, where Σ_K and Σ_I are sets of KDs and IDs respectively. Suppose that, during the construction of $chase_\Sigma(D)$, we apply the FD chase rule to two facts θ_1, θ_2 in $chase_\Sigma(D)$; then all the applications of the FD chase rule that are done in consequence of the first one involve facts that are at level greater or equal than* $\max(level(\theta_1), level(\theta_2)) - \delta$*, where $\delta = |\Sigma| \cdot W!$.*

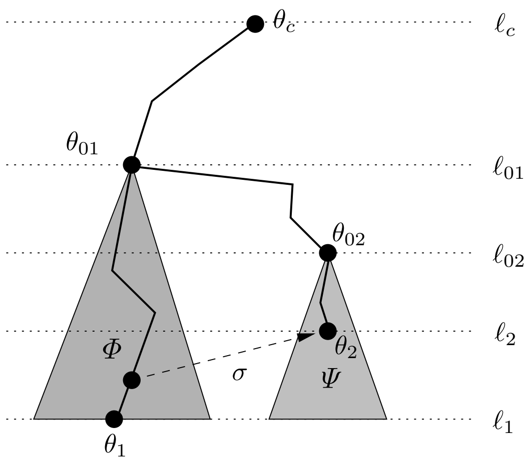

Fig. 4. Figure for the proof of Lemma 4

Proof. Let $key(R) = \{k\}$ be a KD in Σ_K, $\theta_1 = R(\alpha_1, \ldots, \alpha_{k-1}, c, \alpha_{k+1}, \ldots, \alpha_n)$, and $\theta_2 = R(\beta_1, \ldots, \beta_{k-1}, c, \beta_{k+1}, \ldots, \beta_n)$. We refer the reader to Figure 4, that shows the chase graph for $chase_\Sigma(D)$ (higher levels are below in the figure). We assume that $\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n$ are fresh constants in Γ_f (at the end of the proof we shall consider the case where one of the two facts θ_1, θ_2 is in D). In the following we shall not consider IDs and FDs regarding attributes, since they are acyclic and have a marginal role in the construction of the chase. Also, we assume $\ell_1 = level(\theta_1) \geq level(\theta_2) = \ell_2$; this is done without loss of generality, since the other case is symmetric to this one. Since θ_1 and θ_2 agree on the key, we need to turn α_i into β_i for all i such that $1 \leq i \leq n$ and $i \neq k$; in fact, since

θ_2 was generated earlier that θ_1, its fresh constants have higher lexicographic rank; as a consequence, θ_1 is turned into θ_2, so that the arc incoming into θ_1 becomes a cross-arc incoming into θ_2, labelled with the ID σ in Figure 4. Since c appears both in θ_1 and θ_2, it must have appeared for the first time al level ℓ_c, in the fact θ_c; then it propagated in the chase to θ_1 and θ_2.

Let θ_{01} and θ_{02} be the facts where the α_i and β_i appear for the first time, respectively; notice that in general, when fresh constants appear for the first time at levels greater than 0, they occupy all positions in a fact, except one, that contains a constant appearing at the previous level. In the figure, the level ℓ_{01} of θ_{01} is lower than the level ℓ_{02} of θ_{02}: the other case is treated analogously, since it is symmetrical. The shaded subgraphs Φ, Ψ in Figure 4 are the subtrees (considering ordinary arcs only) rooted in θ_{01} and θ_{02} repectively. Therefore: in Φ we find constants $\alpha_1, \ldots, \alpha_{k-1}, c, \alpha_{k+1}, \ldots, \alpha_n$, plus fresh constants introduced in Φ for the first time; in Ψ we find constants $\beta_1, \ldots, \beta_{k-1}, c, \beta_{k+1}, \ldots, \beta_n$, plus fresh constants introduced in Ψ for the first time; moreover, α_1 and β_i appear *only* in Φ, Ψ respectively. By Lemma 3, $\ell_1 - \ell_{01} \leq \delta$, where $\ell_1 = \mathit{level}(\theta_1)$; therefore, changing α_i into β_1 ($1 \leq i \leq n$ and $i \neq k$) affects portions of the chase that are less than δ levels far from θ_1; moreover, applications of the FD chase rule on facts in $\Phi \cup \Psi$ will clearly affect only facts in $\Phi \cup \Psi$ itself. Finally, in the case where θ_{01} (or θ_{02}) is in D, Lemma 3 show immediately that the thesis holds. This proves the claim. □

Lemma 5. *Let Q_1, Q_2 be two conjunctive queries, $\Sigma = \Sigma_K \cup \Sigma_I$ a set of CDs, where Σ_K and Σ_I are sets of KDs and IDs respectively. If there exists a homomorphism μ sending each constant of Γ to itself, and mapping $\mathit{body}(Q_2)$ to facts of $\mathit{chase}_\Sigma(\mathit{fr}(\mathit{body}(Q_1)))$ and $\mathit{head}(Q_2)$ to $\mathit{fr}(\mathit{head}(Q_1))$, then there exists another homomorphism μ' having the same properties, that sends all the facts of $\mathit{body}(Q_2)$ to facts of the database obtained by constructing the first $(|Q_2| + 1) \cdot \delta$ levels of $\mathit{chase}_\Sigma(\mathit{fr}(\mathit{body}(Q_1)))$, where $\delta = |\Sigma| \cdot W!$, according to the given procedure of applications of the chase rules (Definition 2).*

Proof. The proof descends straightforwardly from Lemmata 2 and 4, as discussed above. □

The following theorem is a direct consequence of the previous lemma.

Theorem 3. *Let Q_1, Q_2 be two conjunctive queries, $\Sigma = \Sigma_K \cup \Sigma_I$ a set of CDs, Where Σ_K and Σ_I are sets of KDs and IDs respectively. Checking whether $Q_1 \subseteq_\Sigma Q_2$ is decidable, and can be done by constructing the first $(|Q_2|+1) \cdot |\Sigma| \cdot W!$ levels of $\mathit{chase}_\Sigma(\mathit{fr}(\mathit{body}(Q_1)))$, and checking for the existence of a homomorphism μ' as in Theorem 5.*

As for the complexity of the algorithm for checking a containment $Q_1 \subseteq_\Sigma Q_2$ in case Σ is a set of CDs, we first focus on the complexity w.r.t. $|Q_1|$ and $|Q_2|$ (number of atoms of Q_1 and Q_2 respectively); it is easy to see that our algorithm can be run in time polynomial in $|Q_1|$, and exponential in $|Q_2|$. This because the depth of our finite segment of chase does not depend on $|Q_1|$, and it is linear in $|Q_2|$. The algorithm is also double exponential in W.

The complexity w.r.t. $|Q_1|$ is especially important because, when considering the correspondence between query containment and query answering over a knowledge base or incomplete data [1, 4], Q_1 plays the role of the data, and the complexity w.r.t. $|Q_1|$, called *data complexity*, is highly relevant, since the size of the data is usually much larger that that of the schema. Though decidability of query containment in our case could be proved from the results in [6], our techniques provides a better insight on the complexity of the problem, as discussed in the following section.

5 Discussion

In this paper we have presented a conceptual model based on the ER model, and we have given its semantics in terms of the relational database model with integrity constraints. We have considered conjunctive queries expressed over conceptual schemata, and we have shown that containment of such queries is decidable by means of an algorithm that performs better than all the known ones.

Containment of queries is a fundamental topic in database theory [7, 6, 11, 12]. [3] deals with conceptual schemata in the context of data integration, but the cardinality constraints are more restricted than in our approach. Another work that deals with dependencies similar to those presented here is [5], however the is-a relation among relationships is not considered in it. Also [15] addresses the problem of query containment using a formalism for the schema that is more expressive than the one presented here; however, the problem here is proved to be coNP-hard. In [6], the authors address the problem of query containment for queries on schemata expressed in a formalism that is able to capture our EER model; in this work it is shown that checking containment is decidable and its complexity is exponential in the number of variables and constants of Q_1 and Q_2, and double exponential in the number ov existentially quantified variables that appear in a cycle of the *tuple-graph* of Q_2 (we refer the reader to the paper for further details). Since the complexity is studied by encoding the problem in a different logic, it is not possible to analyse in detail the complexity w.r.t. $|Q_1|$ and $|Q_2|$, which by the technique of [6] is in general exponential. Our work provides a more detailed analysis of the computational cost, showing a lower complexity w.r.t. $|Q_1|$.

The complexity results about query containment are directly applicable in certain cases of answering queries on incomplete databases or knowledge bases, and also in data integration under constraints; still, effective and efficient algorithms are yet to be developed. As for future work, we plan to tackle the problem of answering queries over data integration sytems where the schema is expressed in the EER model, in a way that is similar to the one followed in [3].

Acknowledgments. This work was partly supported by the EU project TONES (IST-007603), and by the Italian national project MAIS. I wish to warmly thank Maurizio Lenzerini, who suggested me to investigate the topic of this paper; I am also grateful to Leopoldo Bertossi, Diego Calvanese and Michael Kifer for valuable comments about this material.

References

1. Serge Abiteboul and Oliver Duschka. Complexity of answering queries using materialized views. In *Proc. of PODS'98*, pages 254–265, 1998.
2. Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison Wesley Publ. Co., 1995.
3. Andrea Calì, Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Accessing data integration systems through conceptual schemas. In *Proc. of ER 2001*, pages 270–284, 2001.
4. Andrea Calì, Domenico Lembo, and Riccardo Rosati. On the decidability and complexity of query answering over inconsistent and incomplete databases. In *Proc. of PODS 2003*, pages 260–271, 2003.
5. Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. DL-Lite: Tractable description logics for ontologies. In *Proc. of AAAI 2005*, pages 602–607, 2005.
6. Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. On the decidability of query containment under constraints. In *Proc. of PODS'98*, pages 149–158, 1998.
7. Edward P. F. Chan. Containment and minimization of positive conjunctive queries in OODB's. In *Proc. of PODS'92*, pages 202–211, 1992.
8. Ashok K. Chandra and Philip M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *Proc. of STOC'77*, pages 77–90, 1977.
9. P. P. Chen. The Entity-Relationship model: Toward a unified view of data. *ACM Trans. on Database Systems*, 1(1):9–36, March 1976.
10. E. F. Codd. A relational model of data for large shared data banks. *Comm. of the ACM*, 13(6):377–387, 1970.
11. David S. Johnson and Anthony C. Klug. Testing containment of conjunctive queries under functional and inclusion dependencies. *J. of Computer and System Sciences*, 28(1):167–189, 1984.
12. Phokion G. Kolaitis and Moshe Y. Vardi. Conjunctive-query containment and constraint satisfaction. In *Proc. of PODS'98*, pages 205–213, 1998.
13. David Maier, Alberto O. Mendelzon, and Yehoshua Sagiv. Testing implications of data dependencies. *ACM Trans. on Database Systems*, 4:455–469, 1979.
14. David Maier, Yehoshua Sagiv, and M. Yannakakis. On the complexity of testing implications of functional and join dependencies. *J. of the ACM*, 28(4):680–695, 1981.
15. Maria Magdalena Ortiz de la Fuente, Diego Calvanese, Thomas Eiter, and Enrico Franconi. Data complexity of answering conjunctive queries over SHIQ knowledge bases. Technical report, Free University of Bozen-Bolzano, 2005. http://arxiv.org/abs/cs.LO/0507059/.
16. Moshe Vardi. Inferring multivalued dependencies from functional and join dependencies. *Acta Informatica*, 19:305–324, 1983.