Journal of Computer and System Sciences ••• (••••) •••–•••

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Inclusion dependencies and their interaction with functional dependencies in SQL ☆

Henning Koehler [a], Sebastian Link [b,*]

[a] School of Engineering & Advanced Technology, Massey University, New Zealand
[b] Department of Computer Science, University of Auckland, New Zealand

ARTICLE INFO

Article history:
Received 17 December 2015
Received in revised form 19 August 2016
Accepted 14 November 2016
Available online xxxx

Keywords:
Axiomatization
Chase
Complexity
Functional dependency
Inclusion dependency
Null
Partial semantics
Simple semantics
Undecidability
SQL

ABSTRACT

Driven by the SQL standard, we investigate simple and partial inclusion dependencies (INDs) with not null constraints. Implication of simple INDs and not null constraints is not finitely axiomatizable. We propose not null inclusion dependencies (NNINDs) that subsume simple and partial INDs, are finitely axiomatizable and PSPACE-complete to decide. NNINDs are fixed parameter-tractable in their arity, typed acyclic NNINDs are NP-hard, and tree-like NNINDs are decidable in linear time. We use a chase to decide implication for functional dependencies and acyclic NNINDs in exponential time, and identify a liberal condition that guarantees no interaction between functional dependencies and acyclic (NN)INDs.

1. Introduction

Motivation. Domain, entity and referential integrity form the three most fundamental classes of integrity rules, already proposed by Codd in his seminal paper [20]. Referential integrity is enforced by foreign keys and, more generally, inclusion dependencies (INDs). INDs empower us to specify which data must be duplicated in what relations, they can express referential integrity at the logical level and are an invaluable tool for classical data management tasks, including conceptual data modeling [18], schema design [28,42,46], and query answering [14,34,51]. An IND $R[X] \subseteq S[Y]$ with attribute sequences $X = [A_1, \ldots, A_n]$ and $Y = [B_1, \ldots, B_n]$ on relation schemata R and S, respectively, is satisfied by a relational database with R-relation r and S-relation s, if for every R-tuple t there is some S-tuple t' such that for all $i = 1, \ldots, n$, the value $t(A_i)$ matches the value $t'(B_i)$. An example IND in the TPC-C schema[1] is

$$\textsc{Order}[\text{c\_id, d\_id, w\_id}] \subseteq \textsc{Customer}[\text{c\_ id, d\_id, w\_id}]$$

☆ The submission is an extended version of our conference paper *Inclusion dependencies reloaded*, presented at the 24th ACM International Conference on Information and Knowledge Management (CIKM 2015), Melbourne, Australia. The research is supported by the Marsden Fund Council from Government funding, administered by the Royal Society of New Zealand.

* Corresponding author.
E-mail addresses: h.koehler@massey.ac.nz (H. Koehler), s.link@auckland.ac.nz (S. Link).

[1] http://www.tpc.org/tpcc/.

http://dx.doi.org/10.1016/j.jcss.2016.11.004
0022-0000/

which states that every order references some customer c_id who placed the order in a district d_id of a warehouse w_id. As {c_id, d_id, w_id} forms a key over CUSTOMER, this IND is even a foreign key.

In the context of the relational model, INDs and their core computational problem of deciding implication have received dedicated interest, see for example [15,21,22,40,41,49,50]. The $\mathcal{C}$-implication problem for a class $\mathcal{C}$ of data dependencies is to decide whether for every given database schema and every given set $\Sigma \cup \{\varphi\}$ of dependencies from $\mathcal{C}$ over the schema, every database instance that satisfies all dependencies in Σ also satisfies φ.

The $\mathcal{C}$-implication problem has always been the central problem in logic [8], and has several important motivations in database theory and practice. For example, solutions to the implication problem allow database designers to decide whether some data dependency φ must be specified on top of the set Σ of dependencies that have already been specified on the given database schema in order to guarantee data integrity under updates. Similarly, they allow designers to compute sets Σ of data dependencies in which no element $\varphi \in \Sigma$ is enforced redundantly, thereby ensuring a minimal overhead in enforcing data integrity.

In schema normalization, such as 3NF synthesis or BCNF decomposition, it must be decided whether a sub-schema satisfies the given normal form condition, which is only achieved by deciding the associated implication problem [7,26,42]. In query optimization, deciding implication makes it possible to eliminate superfluous joins or `DISTINCT` clauses [1,29], for example. In database security, the implication problem of data dependencies is central because implied data dependencies enable users to bypass access control and infer confidential information [6,27]. Renewed interest in INDs comes from modern applications such as data cleaning and schema matching [9,45], data integration [10], data profiling [4,19,56], as well as query answering and the semantic web [13,29,44,52,55].

Surprisingly, INDs have only received little attention in the context of the industry standard SQL, which subsumes null-free relations as the idealized special case where all attributes of the schema have been declared not null, that is, the null marker $\perp$ must not feature in any of the table columns. In fact, the SQL standard proposes a 'simple' and a 'partial' semantics for INDs. Both semantics apply the definition from the null-free relational case above, but they differ in which tuples t they consider. For an IND $R[X] \subseteq_p S[Y]$ under partial semantics, all tuples t are considered, and $t(A_i)$ matches $t'(B_i)$ if and only if $t(A_i) = \perp$ or $t(A_i) = t'(B_i)$ holds. For an IND $R[X] \subseteq_s S[Y]$ under simple semantics there is a distinction between R-tuples t which are total on X, that is, where for all $i = 1, \ldots, n$, the value $t(A_i)$ is different from $\perp$, and R-tuples t which are partial on X, that is, not total on X. Only X-total tuples t are considered, and $t(A_i)$ matches $t'(B_i)$ if and only if $t(A_i) = t'(B_i)$ holds. We shall refer to INDs under simple (partial) semantics as simple (partial) INDs.

For an example, consider a database where the only tuple t of the database occurs in the ORDER-relation and has projection $[\perp, d_1, w_1]$ on $[c\_id, d\_id, w\_id]$. This database satisfies the simple IND

$$\text{ORDER[c\_id, d\_id, w\_id]} \subseteq_s \text{CUSTOMER[c\_id, d\_id, w\_id]},$$

but violates the partial IND

$$\text{ORDER[c\_id, d\_id, w\_id]} \subseteq_p \text{CUSTOMER[c\_id, d\_id, w\_id]}.$$

It is well-known that simple semantics for foreign keys is natively supported by all SQL implementations, but partial semantics is not natively supported by any SQL implementation [31,48].

On the one hand, it is not surprising that the implication problems for INDs under simple and partial semantics are different from one another. For example, the partial IND above implies the partial IND

$$\text{ORDER[d\_id, w\_id]} \subseteq_p \text{CUSTOMER[d\_ id, w\_id]},$$

but the simple IND above does not imply the simple IND

$$\text{ORDER[d\_id, w\_id]} \subseteq_s \text{CUSTOMER[d\_ id, w\_id]}.$$

On the other hand, it is surprising that the implication problem has only been investigated under partial semantics, and only under the assumption that the null marker can occur in every column [38]. In fact, we will see that the implication problem for INDs under partial semantics and not null constraints enjoys the same axiomatization as INDs on null-free relational databases [15]. In addition, for simple INDs alone (i.e., without not null constraints) we will establish an axiomatization that results from the axiomatization for partial INDs by omitting the projection rule.

However, things already change dramatically when we consider simple INDs in combination with not null constraints. In this case, we will show that the associated implication problem is not k-ary axiomatizable for any finite k, i.e., cannot be axiomatized by inference rules with at most k premises, and in particular not by any finite set of rules. This result is rather discouraging for database practice in which simple semantics is natively supported, but not partial semantics [31,48]. Concise axiomatizations aid in human understanding of constraint interaction, and critical applications such as schema design, integrity enforcement and query processing. We emphasize the importance of the ability to reason about simple and partial INDs under not null constraints by the following example from database design.

Example 1. Consider the following relation schemata, in which attributes A that do not permit null values are denoted by A[NN]:

Please cite this article in press as: H. Koehler, S. Link, Inclusion dependencies and their interaction with functional dependencies in SQL, J. Comput. Syst. Sci. (2016), http://dx.doi.org/10.1016/j.jcss.2016.11.004