On the Computation of Relational View Complements

JENS LECHTENBÖRGER and GOTTFRIED VOSSEN
University of Münster, Münster, Germany

Views as a means to describe parts of a given data collection play an important role in many database applications. In dynamic environments where data is updated, not only information provided by views, but also information provided by data sources yet *missing* from views turns out to be relevant: Previously, this missing information has been characterized in terms of *view complements*; recently, it has been shown that view complements can be exploited in the context of data warehouses to guarantee desirable warehouse properties such as independence and self-maintainability. As the complete source information is a trivial complement for any view, a natural interest for "small" or even "minimal" complements arises. However, the computation of minimal complements is still not very well understood. In this article, it is shown how to compute reasonably small (and in special cases even minimal) complements for a large class of relational views.

Categories and Subject Descriptors: H.2.4 [**Database Management**]: Systems—*relational databases*; H.2.7 [**Database Management**]: Database Administration—*data warehouse and repository*

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Relational algebra, views, view complements, minimal complements, data warehouses, self-maintainability

1. INTRODUCTION

The notion of a *view* as a means to describe parts of a given data collection plays an important role in many database applications. For example, views may be used to restrict the amount of information that a user has to care about or is allowed to access, or views can be used to formalize the integration of (part of the) information provided by multiple data sources. In dynamic environments where data is updated, not only information provided by views, but also

A preliminary version of this article, showing some of the results in considerably condensed form, appeared in *Proceedings of the 21st Annual ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, ACM, New York, 2002, pp. 142–149.
Authors' address: Department of Information Systems, University of Münster, Leonardo-Campus 3, D-48149 Münster, Germany; email: {lechten,vossen}@helios.uni-muenster.de.

information provided by data sources yet *missing* from views, turns out to be relevant: In Bancilhon and Spyratos [1981], this missing information has been characterized in terms of *view complements* and used to translate updates on views to updates on base relations. Intuitively, a complement of a set V of views is another set C of views such that the base relations can be reconstructed from V and C. Recently, it has been shown [Laurent et al. 2001] that view complements can be exploited in the context of data warehouses to guarantee desirable warehouse properties such as independence and self-maintainability. As the complete source information is always a trivial complement for any view, a natural interest for "small" or even "minimal" complements arises. However, the computation of minimal complements is not too well understood. In this article, we show how to compute reasonably small (and in special cases even minimal) complements for monotonic relational views, where the size of complements is measured in terms of their information content.

To give the reader an idea of the improvements of our approach towards the computation of "small" complements compared to the previous methods [Cosmadakis and Papadimitriou 1984; Laurent et al. 2001] consider the following example:

Example 1.1. Consider a single base relation R with attributes A, B, and C, where A is the key, as well as views V_1 and V_2 over R defined as follows: $V_1 =_{df} \pi_{AB}(R)$ and $V_2 =_{df} \sigma_{\phi}(R)$.[1] Now, a complement of $\{V_1, V_2\}$ is a set C of views such that R can be computed from the views in $\{V_1, V_2\}$ and C.

The approach described in Cosmadakis and Papadimitriou [1984], which deals only with projection views, ignores V_2. Moreover, a minimal complement of V_1 according to Cosmadakis and Papadimitriou [1984] is a second projection $\pi_X(R)$ where X is a *minimal subset* of R's attributes such that R can be computed by the join $V_1 \bowtie \pi_X(R)$. Clearly, we hence obtain complement $C_1 = \{\pi_{AC}(R)\}$.

On the other hand, the approach described in Laurent et al. [2001], where every complementary view is restricted to have the same schema as some base relation, does not make use of the information provided by projection view V_1. Without going into details, we note that for the set of views $\{V_1, V_2\}$ Laurent et al. [2001] derive the complement $C_2 = \{R \setminus V_2\}$, which allows to compute R as $V_2 \cup (R \setminus V_2)$.

By contrast, our approach makes use of *both* views V_1 and V_2 to reduce the size of the complement. In a sense, we present a combination of the approaches in Cosmadakis and Papadimitriou [1984] and Laurent et al. [2001], as our complements extend the ones given in Laurent et al. [2001] by allowing projections given in Cosmadakis and Papadimitriou [1984]. Indeed, Theorem 3.22 (see Section 3) produces complement $C = \{C_R^{AB}, C_R^{AC}\}$ for the set of views $\{V_1, V_2\}$, where

$$C_R^{AB} =_{df} \pi_{AB}(R) \setminus (\pi_{AB}(V_1) \cup \pi_{AB}(V_2))$$
$$C_R^{AC} =_{df} \pi_{AC}(R) \setminus \pi_{AC}(V_2).$$

[1]The symbol $=_{df}$ associates a view name with a defining expression, cf. Section 2.1.

We note that $\pi_{AB}(V_1) = V_1 = \pi_{AB}(R)$, hence C_R^{AB} is constantly empty and can therefore be removed from C without loss of information; moreover, this fact can be detected using the techniques of Sagiv and Yannakakis [1980]. Based on this remark, it should be clear that our complement C is intuitively smaller than both previous complements C_1 and C_2. In fact, it can be shown that C is indeed smaller than C_1 and C_2 with respect to information content (as formalized by the view ordering $<_{ic}$; see Definition 2.4). Finally, R can be computed from $\{V_1, V_2\}$ and C by $V_2 \cup (V_1 \bowtie C_R^{AC})$.

The results presented in this article extend the approach towards the computation of view complements previously presented in Laurent et al. [2001] in the following nontrivial ways:

—In Laurent et al. [2001], the computation of complements for PSJ views, that is, views defined by join, followed by selection, followed by projection, was studied, whereas we additionally consider renaming and union.

—The size of complements was measured based on query containment in Laurent et al. [2001], whereas we use information content for this purpose, which allows (a) to compare larger classes of candidate complements and (b) leads to uniformly smaller complements.

—The costs for computing complements according to Laurent et al. [2001] are exponential in the size of schema information, whereas they are polynomial according to our new approach.

The remainder of this article is organized as follows. In Section 2, we introduce the basic notions to be used throughout this text. In Section 3, we present the main results of this article, namely expressions for the computation of "small" complements for sets of views defined by renaming, join, selection, projection, and union. We also study the complexity of constructing the complementary expressions and show it to be polynomial in the size of schema information. We discuss a variety of related work and also applications in Section 4, and we conclude in Section 5.

2. BACKGROUND AND NOTATION

In this article we consider relational databases and relational views with set semantics. We assume the reader to be familiar with the relational algebra (see, e.g., Abiteboul et al. [1995], Silberschatz et al. [2002], and Vossen [2000]) and recall basic notions only briefly while fixing our notation.

2.1 Relational Databases and Views

A relation schema R is defined by a name, a set of attributes, which is denoted by $attr(R)$, and a key dependency. We use $key(R)$ ($\subseteq attr(R)$) to denote the unique key of relation schema R, and we assume that no functional dependencies except those implied by the key dependency hold for a given relation schema. An instance r of relation schema R is a set of tuples over $attr(R)$ that satisfies the associated key dependency. A database schema D is defined by a name

and a set of relation schemata.[2] We may identify D with the set of its relation schemata, that is, we may write $D = \{R_1, \ldots, R_n\}$ if $R_1, \ldots, R_n$ are the relation schemata of D. An instance of database schema D is a function d that maps each relation schema R of D to an instance of R, denoted by $d(R)$, such that the key dependencies are satisfied.

Given a database schema D, a relational view V over D is defined by a name and a relational expression, where only names of relation schemata of D occur. We assume that every relational expression is composed of relation schema names and the following basic operations: renaming (ρ_f, where f is an injective function over attribute names), selection (σ_ϕ, where ϕ is a selection condition), projection (π_X, where X is a set of attribute names), join ($\bowtie$), union ($\cup$), and difference ($\setminus$). We use the notation *View-name* $=_{df}$ *Expression* to introduce a view named *View-name* whose defining relational expression is *Expression*.

We note that each view is associated with a set of attributes, namely the attributes of the relation computed by the view's relational expression. Therefore, a view can also be seen as a relation schema. (For simplicity, we assume that no dependencies are associated with views.) In analogy to the notation used for relation schemata, we use $attr(V)$ to refer to the set of attributes associated with view V.

Let D be a database schema, let d be an instance of D, and let V be a view over D. Then the instance of V in d, denoted by $V(d)$, is the relation over $attr(V)$ that is computed by evaluating the relational expression associated with V in database instance d.

We next recall the definitions of containment and equivalence of relational expressions (cf. Abiteboul et al. [1995] and Vossen [2000]).

Definition 2.1. Let D be a database schema. Let $\mathcal{E}_1$ and $\mathcal{E}_2$ be relational expressions over D with $attr(\mathcal{E}_1) = attr(\mathcal{E}_2)$.

(1) $\mathcal{E}_1$ is *contained in* $\mathcal{E}_2$, denoted by $\mathcal{E}_1 \preceq_{qc} \mathcal{E}_2$, if for all instances d of D we have $\mathcal{E}_1(d) \subseteq \mathcal{E}_2(d)$.[3]

(2) $\mathcal{E}_1$ and $\mathcal{E}_2$ are *equivalent*, denoted by $\mathcal{E}_1 \approx \mathcal{E}_2$, if $\mathcal{E}_1 \preceq_{qc} \mathcal{E}_2$ and $\mathcal{E}_2 \preceq_{qc} \mathcal{E}_1$.

Since queries and views are defined by relational expressions, Definition 2.1 immediately applies to queries and views. Similarly, the following notions are defined for expressions but apply to queries and views as well.

A relational expression $\mathcal{E}$ (or a view or a query) over database schema D is *monotonic* if we have $\mathcal{E}(d_1) \subseteq \mathcal{E}(d_2)$ for all instances d_1 and d_2 of D such that $d_1 \subseteq d_2$. (An instance d_1 is contained in an instance d_2, denoted by $d_1 \subseteq d_2$, if we have $d_1(R) \subseteq d_2(R)$ for all relation schemata R of D.) Furthermore, it is well known [Abiteboul et al. 1995; Sagiv and Yannakakis 1980] that equivalence is decidable for relational expressions involving renaming, selection with positive conjunctive selection condition, projection, join, and union and that

[2]We note that the computation of *minimal* complements in the presence of inter-relational dependencies such as inclusion dependencies lies outside the scope of this paper. Hence, except for some remarks at the end of Section 4 we assume that there are no inter-relational dependencies.

[3]The subscript qc stands for "query containment." In Section 2.2, we introduce a second ordering of views with respect to their information content, denoted by $\preceq_{ic}$; cf. Definition 2.4.

such expressions are monotonic. Moreover, every expression involving renaming, selection with positive conjunctive selection conditions, projection, join, and union is equivalent to a so-called UPSJR expression, where renaming operations are applied first (if any), followed by join (if any), then selection (if any), then projection (if any), and finally union (if any). Finally, PSJR (respectively SJR) expressions form the subclass of UPSJR expressions that do not contain union (respectively union and projection). Note that PSJR expressions are exactly the *conjunctive* expressions [Abiteboul et al. 1995].

2.2 View Complements

A set V of views over database schema D expresses some, but usually not all of the information contained in D. Informally, any set C of views over D that expresses the information "missing" in V with respect to D is called a *complement* of V with respect to D. Bancilhon and Spyratos [1981] have introduced the notion of complement in a formal framework where views are defined by *arbitrary* functions. In our setting of *relational* views, we have the following definition:

Definition 2.2. Let V be a set of views over $D = \{R_1, \ldots, R_n\}$. A *complement* of V (with respect to D) is a set C of views over D such that the following holds: For every $i = 1, \ldots, n$, there exists a relational expression $\mathcal{E}_i$ over views in $V \cup C$ *only* such that $R_i \approx \mathcal{E}_i$. In this case, we call the set of equivalences $R_i \approx \mathcal{E}_i$, $1 \leq i \leq n$, the *view inverse defined by* C.

If C contains exactly one view C_R per base relation schema $R \in D$ such that $attr(C_R) = attr(R)$, then we call C a *tally* complement.

Roughly speaking, C is a complement of V if every instance of D can be computed from the corresponding instances of V and C. Clearly, a complete copy of all database relations is a trivial complement for every set of views.

Example 2.3. Consider a banking application where two relation schemata contain account information, namely `AccStatus` dealing with current states of accounts and `AccProfit` about measures rating the profitability of accounts:

—$attr$(AccStatus) = {AccountID, Day, Balance, NoOfTransactions, Creditlimit, Interest},
—$attr$(AccProfit) = {AccountID, Day, Turnover, Profitability}.

Let D = {`AccStatus`, `AccProfit`}, and consider the following view *Facts* over D:

$$Facts =_{df} \text{AccStatus} \bowtie \text{AccProfit}$$

Let $V = \{Facts\}$, and consider the views

—$C_{AS} =_{df}$ AccStatus,
—$C_{AP} =_{df}$ AccProfit,

which provide copies of relation schemata in D. Clearly, every instance of D can be computed from instances of V and $C = \{C_{AS}, C_{AP}\}$. Hence, C is a complement of V with respect to D. Moreover, C contains exactly one complementary view

for each base relation such that the sets of attributes of complementary view and base relation are the same. Hence, C is a tally complement.

As the complete source information is a trivial complement for any given view, a natural interest for "small" or even "minimal" complements arises. Indeed, the following ordering of views was proposed by Bancilhon and Spyratos [1981] to compare the information content, and thereby the size, of views.

Intuitively, a set V_1 of views is smaller than a set V_2 of views, if V_1 does not distinguish more database instances than V_2 does. Formally, we have:

Definition 2.4. Let V_1 and V_2 be sets of views over D. V_1 is *smaller than* V_2, denoted by $V_1 \leq_{ic} V_2$, if $V_2(d_1) = V_2(d_2)$ implies $V_1(d_1) = V_1(d_2)$ for all instances d_1 and d_2 of D. V_1 is *strictly smaller than* V_2, denoted by $V_1 <_{ic} V_2$, if $V_1 \leq_{ic} V_2$ and there are instances d_1 and d_2 of D such that $V_1(d_1) = V_1(d_2)$ and $V_2(d_1) \neq V_2(d_2)$.

We recall from Bancilhon and Spyratos [1981] that the largest (sets of) views with respect to the above view ordering are those whose associated functions over instances are injective, such as complete copies of all relations, whereas the smallest (sets of) views are those with constant functions, such as constantly empty views.

Example 2.5. We continue our banking application and note that all attributes of the relation schemata `AccStatus` and `AccProfit` occur in view *Facts*. Now consider the following views over D:

$$C'_{AS} =_{df} \text{AccStatus} \backslash \pi_{attr(\text{AccStatus})}(\mathit{Facts})$$
$$C'_{AP} =_{df} \text{AccProfit} \backslash \pi_{attr(\text{AccProfit})}(\mathit{Facts})$$

Then $C' = \{C'_{AS}, C'_{AP}\}$ is a tally complement of V with respect to D. Indeed, the view inverses, which allow to compute instances of D from instances of V and C', are given as follows:

$$\text{AccStatus} \approx C'_{AS} \cup \pi_{attr(\text{AccStatus})}(\mathit{Facts})$$
$$\text{AccProfit} \approx C'_{AP} \cup \pi_{attr(\text{AccProfit})}(\mathit{Facts})$$

We clearly have $C' \leq_{ic} C$, as C preserves all information from D. Moreover, we even have $C' <_{ic} C$. Indeed, consider the following instances d_1 and d_2 of D. Let d_1 be the empty instance, and let d_2 be an instance where each of both relations contains exactly one tuple for the same account on the same day, say, for an account whose `AccountID` is 1 on January 1, 2000.

We observe that $\mathit{Facts}(d_1)$ is empty, whereas $\mathit{Facts}(d_2)$ contains a single tuple concerning the account whose `AccountID` is 1. Then we obtain $C'(d_1) = C'(d_2)$ (as both views are empty in both instances). Moreover, we have $C(d_1) \neq C(d_2)$ (as both views are empty in d_1 whereas each of them contains a single tuple in d_2). Consequently, we have $C' <_{ic} C$.

The following proposition, which goes back to Bancilhon and Spyratos [1981] (and is used in Laurent et al. [2001] as well), states a fundamental property of complements. Indeed, given a set V of views over D, each complement C of V sets up a one-to-one mapping from instances of D to instances of $V \cup C$.

PROPOSITION 2.6. *Let V and C be sets of views over D. If C is a complement of V, then $d \neq d'$ implies $(V \cup C)(d) \neq (V \cup C)(d')$ for all instances d and d' of D.*

The complements to be computed in this article exhibit a particular syntactic form, where UPSJR expressions play a central role. The following definition introduces this kind of complements, called *super-key based* complements.

Definition 2.7. Let V be a set of views over D. Then C is a *super-key based complement* of V, if C is a complement of V such that the following properties are all satisfied:

(1) The set C contains only views of the form
$$C_R^X =_{df} \mathcal{E}_R^X,$$
where $\mathcal{E}_R^X$ is equivalent to an expression of the form $\pi_X(R) \backslash E_R^X$ for $R \in D$, a super-key X of R, and a UPSJR expression E_R^X over V. In particular, for every $R \in D$ and every super-key X of R, the set C contains at most one view of this form.

Given $R \in D$ we say that the views C_R^X are the *(super-key based) complementary views* for relation schema R and we use $C[R]$ to denote this subset of C.

(2) The view inverse for $R \in D$, is equivalent to an expression of the form
$$\mathcal{I}_R^{X_1}\left(C_R^{X_1}, V\right) \bowtie \cdots \bowtie \mathcal{I}_R^{X_k}\left(C_R^{X_k}, V\right),$$
where $C[R] = \{C_R^{X_1}, \ldots, C_R^{X_k}\}$, $k \geq 1$, and $\mathcal{I}_R^{X_i}$ is a relational expression over $C_R^{X_i}$ and V such that $\pi_{X_i}(R) \approx \mathcal{I}_R^{X_i}(C_R^{X_i}, V)$, $1 \leq i \leq k$.

Example 2.8. In our banking application, we note that the views of the above complements C and C' satisfy the first condition of Definition 2.7. Moreover, as there is exactly one complementary view for each relation schema, the join operation can be removed from the second condition of Definition 2.7; the resulting condition is satisfied by the above view inverses. Hence, C and C' are super-key based complements.

The following comments concerning Definition 2.7 are apt. First, this definition generalizes the notion of complement introduced in Laurent et al. [2001] by allowing complementary views to be defined on super-keys of relation schemata. In contrast, complements of Laurent et al. [2001] are tally complements. Next, following Sagiv and Yannakakis [1980] query containment is decidable for UPSJR views, and the tableaux techniques of Sagiv and Yannakakis [1980] may be applied to optimize the UPSJR subexpressions of super-key based complementary views. In particular, given a complementary view $C_R^X =_{df} \pi_X(R) \backslash E_R^X$, these techniques allow us to detect whether $\pi_X(R)$ is contained in E_R^X, that is, whether C_R^X is constantly empty and, hence, not needed for computing base relations.

Finally, we point out that the second condition of Definition 2.7 may appear redundant at first sight. However, the following example exhibits a complement that satisfies only the first but not the second condition of Definition 2.7.

Example 2.9. Let $D = \{R_1, R_2\}$, where $attr(R_1) = attr(R_2)$. Consider views $V = \{V_1, V_2\}$ over D, where $V_1 =_{df} R_1 \cup R_2$ and $V_2 =_{df} R_1 \cap R_2$. Let $C = \{C_{R_1}, C_{R_2}\}$, where $C_{R_1} =_{df} R_1 \backslash V_2$ and $C_{R_2} =_{df} \emptyset$.

Then C is a complement of V, as we have $R_1 \approx V_2 \cup C_{R_1}$ and $R_2 \approx (V_1 \backslash (V_2 \cup C_{R_1})) \cup V_2$. Moreover, C_{R_1} and C_{R_2} satisfy the first condition of Definition 2.7. However, the view inverse for R_2 involves C_{R_1}, which violates the second condition of Definition 2.7. Consequently, C is not a super-key based complement. Finally, R_2 cannot be computed from the symmetric information contained in the views alone. Hence, some complementary information is necessary to compute R_2 using views. As C_{R_1} is the only nonempty complementary view, *every* view inverse for R_2 must involve C_{R_1}.

3. COMPUTATION OF VIEW COMPLEMENTS

In this section, we present our main results concerning the computation of view complements. To this end, we assume that every domain associated with an attribute contains at least two values. We note that this assumption, which is well known in database theory and trivially holds for any real-world database, will be used in the proofs of Lemmata 3.9 and 3.10 to show certain minimality results.

In order to prove minimality of complements, we first present an important technical lemma, which provides a sufficient condition for minimality of certain kinds of complements with respect to the view ordering "$\leq_{ic}$". The lemma relies on the following definition.

Definition 3.1. Let V be a set of views over D, and let C be a tally complement of V. Given an instance d of D, the *instance of D induced by C for d*, denoted by $d_C(d)$, is the instance of D that maps R to relation $C_R(d)$, that is, $d_C(d) = \langle C_{R_1}(d), \ldots, C_{R_n}(d) \rangle$.[4]

Definition 3.1 allows to interpret an instance $C(d)$ of a tally complement in a natural way as induced instance $d_C(d)$ of the database relations. The following lemma uses such induced instances to provide a sufficient condition for minimality of C.

LEMMA 3.2. *Let V be a set of views over D, and let C be a tally complement of V. If*

(1) *$V(d_C(d)) = \emptyset$ for all instances d of D and*
(2) *$d_C(d) = d$ for all instances d of D with $V(d) = \emptyset$,*

then C is a minimal complement with respect to $<_{ic}$.

PROOF. Let C be a tally complement of V. Assume for the sake of a contradiction that C is not minimal, that is, that there is a complement C' of V such that $C' <_{ic} C$. By definition of "$<_{ic}$" there are instances d_1 and d_2 of D such that $C(d_1) \neq C(d_2)$ and $C'(d_1) = C'(d_2)$.

[4]To simplify notation concerning instances of databases as well as of sets of views we use angle brackets to denote instances of databases (and views), if an arbitrary but fixed ordering of relation schemata (and views) is understood.

In this situation, we have $d_1 \neq d_2$ (otherwise, we could not have $C(d_1) \neq C(d_2)$) and $d_C(d_1) \neq d_C(d_2)$ (as we have $C(d_1) \neq C(d_2)$). In addition, it is easy to see that we also have $V(d_1) \neq V(d_2)$. Indeed, as C' is supposed to be a complement, there are inverse expressions to compute d_1 from $V(d_1)$ and $C'(d_1)$ as well as d_2 from $V(d_2)$ and $C'(d_2)$. As we have $d_1 \neq d_2$ and $C'(d_1) = C'(d_2)$, we must have $V(d_1) \neq V(d_2)$ by Proposition 2.6.

By precondition (1) we have $V(d_C(d_1)) = V(d_C(d_2)) = \emptyset$. Thus, by precondition (2), we find $d_C(d_C(d_1)) = d_C(d_1)$ and $d_C(d_C(d_2)) = d_C(d_2)$. The definition of d_C then implies $C(d_C(d_1)) = C(d_1)$ and $C(d_C(d_2)) = C(d_2)$, that is, C does neither distinguish $d_C(d_1)$ from d_1 nor $d_C(d_2)$ from d_2.

Exploiting $C' <_{ic} C$ we deduce that C' does neither distinguish $d_C(d_1)$ from d_1 nor $d_C(d_2)$ from d_2, that is, $C'(d_C(d_1)) = C'(d_1)$ and $C'(d_C(d_2)) = C'(d_2)$. Using $C'(d_1) = C'(d_2)$, we obtain $C'(d_C(d_1)) = C'(d_C(d_2))$.

To summarize, we have constructed instances $d_C(d_1)$ and $d_C(d_2)$ of D such that

(1) $d_C(d_1) \neq d_C(d_2)$,
(2) $V(d_C(d_1)) = V(d_C(d_2))$, and
(3) $C'(d_C(d_1)) = C'(d_C(d_2))$.

Clearly, these three facts contradict Proposition 2.6 (i.e., $V \cup C'$ does not contain enough information to distinguish $d_C(d_1)$ and $d_C(d_2)$), showing that C' is not a complement of V. □

3.1 Minimal Complements of SJR Views

The following theorem states how minimal complements for SJR views can be computed, where the notation f^{-1} is used to denote inverses of renaming functions.

THEOREM 3.3. *Let $V = \{V_1, \ldots, V_m\}$ be a set of SJR views over $D = \{R_1, \ldots, R_n\}$, where each $V_i \in V$, $1 \leq i \leq m$, has the following form:*

$$V_i =_{df} \sigma_{\phi_i}(\rho_{f_{i,i_1}}(R_{i_1}) \bowtie \cdots \bowtie \rho_{f_{i,i_k}}(R_{i_k}))$$

For $R_j \in D$ and $V_i \in V$ let view $\overline{R_j^i}$ be defined as follows, where $attr(\overline{R_j^i}) = attr(R_j)$:

$$\overline{R_j^i} =_{df} \begin{cases} \emptyset & \text{if } R_j \text{ does not occur in } V_i, \\ \bigcup\limits_{R_{i_l} = R_j} \rho_{f_{i,i_l}^{-1}}(\pi_{f_{i,i_l}(attr(R_j))}(V_i)) & \text{otherwise.} \end{cases} \quad (1)$$

For $R_j \in D$, let view $\overline{R_j}$ be defined as follows:

$$\overline{R_j} =_{df} \bigcup_{V_i \in V} \overline{R_j^i} \quad (2)$$

Then, the set of views $C = \{C_{R_1}, \ldots, C_{R_n}\}$, where

$$C_{R_j} =_{df} R_j \backslash \overline{R_j}, \quad 1 \leq j \leq n, \quad (3)$$

is a super-key based and minimal complement of V. The view inverse is defined as follows for $R_j \in D$, $1 \leq j \leq n$:

$$R_j \approx C_{R_j} \cup \overline{R_j}. \tag{4}$$

PROOF. We first show that C is indeed a complement of V, which is super-key based; afterwards we prove minimality. To show that C is a complement, we have to establish the inclusion

$$\overline{R_j^i} \subseteq R_j \tag{5}$$

for $R_j \in D$ and $V_i \in V$. From (5), we then obtain $\overline{R_j} \subseteq R_j$ for all $R_j \in D$, which implies that Equivalence (4) is correct. Since (4) shows how to compute all relations from $V \cup C$, C is a complement of V. Moreover, C is super-key based by construction.

Thus, we need to verify (5). Let $R_j \in D$ and $V_i \in V$. If R_j does not occur in V_i, we have $\overline{R_j^i} =_{df} \emptyset$, and inclusion (5) holds. Therefore, assume that R_j occurs in V_i, say $j = i_l$ for $l \in [1, k]$. We have to show the following inclusion:

$$\rho_{f_{i,i_l}^{-1}}\left(\pi_{f_{i,i_l}(attr(R_j))}(V_i)\right) \subseteq R_j. \tag{6}$$

We first remark that inclusion (6) makes sense due to the assumption that renaming functions are injective; hence, the expressions on the right- and left-hand sides are associated with the same set of attributes, namely $attr(R_j)$. Now, let d be an instance of D, and let

$$t \in \rho_{f_{i,i_l}^{-1}}\left(\pi_{f_{i,i_l}(attr(R_j))}(V_i(d))\right);$$

as we just observed, t is a tuple over $attr(R_j)$. We now use the definitions of operations involved in $\rho_{f_{i,i_l}^{-1}}(\pi_{f_{i,i_l}(attr(R_j))}(V_i))$ to track down the origin of t.

By the definition of renaming, there is a tuple t' over $f_{i,i_l}(attr(R_j))$ in

$$\pi_{f_{i,i_l}(attr(R_j))}(V_i(d))$$

such that $t[A] = t'[f_{i,i_l}(A)]$, for all $A \in attr(R_i)$. Then the definition of projection implies that there is a tuple $t_j \in V_i(d)$ such that $t_j[f_{i,i_l}(attr(R_j))] = t'$. We next analyze the operations in V_i. By the definition of selection, t_j satisfies the condition ϕ_i, and we have

$$t_j \in \rho_{f_{i,i_1}}(R_{i_1}) \bowtie \cdots \bowtie \rho_{f_{i,i_k}}(R_{i_k}).$$

By the definition of join, we then have

$$t_j\left[f_{i,i_l}(attr(R_j))\right] \in \rho_{f_{i,i_l}}(R_{i_l}),$$

which implies $t' \in \rho_{f_{i,i_l}}(R_{i_l})$. The properties of t' finally entail $t \in R_{i_l} (= R_j)$. Thus, inclusion (6) holds, which concludes the proof that C is a complement of V.

Concerning minimality of C we are going to apply Lemma 3.2. Hence, we have to check conditions (1) and (2) of Lemma 3.2 against complements defined by Eq. (3).

(1) Let d be an instance of D, and let $d_C(d)$ be the instance of D induced by C for d. We have to show $V(d_C(d)) = \emptyset$.

Let $R_j \in D$. By inclusion (6), equality $C_{R_j}(d) = R_j(d_C(d))$, and Eq. (3) we obtain

$$\overline{R_j}(d_C(d)) \subseteq R_j(d_C(d)) = C_{R_j}(d) \subseteq R_j(d). \tag{7}$$

As SJR views are monotonic, we find $V_i(d_C(d)) \subseteq V_i(d)$, which implies

$$\overline{R_j}(d_C(d)) \subseteq \overline{R_j}(d). \tag{8}$$

Then $\overline{R_j}(d) \cap C_{R_j}(d) = \emptyset$ and the equality $R_j(d_C(d)) = C_{R_j}(d)$ imply $\overline{R_j}(d) \cap R_j(d_C(d)) = \emptyset$. From the left-hand inclusion of (7), it follows that $\overline{R_j}(d) \cap \overline{R_j}(d_C(d)) = \emptyset$. In view of (8), we then must have $\overline{R_j}(d_C(d)) = \emptyset$, which, by definition of expression $\overline{R_j}$, can only be the case if $V(d_C(d)) = \emptyset$.

(2) Let d be an instance of D such that $V(d) = \emptyset$, and let $R_j \in D$. We have to show $d_C(d) = d$. Obviously, we have $\overline{R_j}(d) = \emptyset$, which implies $C_{R_j} = R_j$, which in turn yields $d_C(d) = d$.

An application of Lemma 3.2 concludes the proof. □

Roughly, given a set V of SJR views, Theorem 3.3 produces a tally complement with one complementary view C_R per base relation R, where the schema of C_R is equal to the schema of R and the instances of C_R contain those tuples of (instances of) R that do not appear in any view in V. Indeed, expression $\overline{R_j^i}$ given by Eq. (1) represents that portion of R_j which can be obtained from view V_i, keeping in mind that R_j can occur multiple times in V_i (under different renaming functions). Then, expression, $\overline{R_j}$ given by Eq. (2) collects those portions for all views in V, and complementary view C_{R_j} given by Eq. (3) covers those tuples of R_j that are not contained in $\overline{R_j}$, that is, that cannot be computed from the views.

It is easily verified that the complementary expressions provided by Theorem 3.3 are a generalization of those given by a similar result of Laurent et al. [2001] for selection and join (without renaming). It is instructive to note that the result of Laurent et al. [2001] shows minimality of the complement with respect to query containment (and that their proof can be generalized to views involving renaming), whereas we additionally have minimality with respect to information content.

Example 3.4. Recall that in our banking application (introduced in Example 2.3) V contains a single view, which is the join of two base relations, that is, which is an SJR view. Hence, Theorem 3.3 is applicable. It is easy to see that Eq. (3) leads to complement C' above. Consequently, C' is a minimal complement.

The next example illustrates the handling of renaming according to Theorem 3.3.

Example 3.5. Consider $D = \{R_1\}$, where R_1(Parent, Child) collects names of parents and their children, and views $V = \{V_1, V_2\}$ over D, where

$$V_1 =_{df} \sigma_{\text{Grandparent=Paul}}(\rho_f(R_1) \bowtie R_1)$$

such that function f renames `Parent` into `Grandparent` and `Child` into `Parent`, that is, V_1 selects grandparents, parents, and children, where the grandparent is Paul, and

$$V_2 =_{df} \sigma_{\text{Child=John}}(R_1)$$

Sample instances d of D, $V_1(d)$ of V_1, and $V_2(d)$ of V_2 are as follows:

R_1	Parent	Child
	Mary	John
	Paul	John
	John	Peter
	Fred	Barney

V_1	Grandparent	Parent	Child
	Paul	John	Peter

V_2	Parent	Child
	Mary	John
	Paul	John

In accordance with Eq. (1), we have

$$\overline{R_1^1} =_{df} \rho_{f^{-1}}(\pi_{\text{Grandparent,Parent}}(V_1)) \cup \pi_{\text{Parent,Child}}(V_1),$$

where f^{-1} is the inverse of f, that is, f^{-1} renames `Parent` into `Child` and `Grandparent` into `Parent`, and

$$\overline{R_1^2} =_{df} \pi_{\text{Parent,Child}}(V_2).$$

Thus, in accordance with Eq. (2), we have

$$\overline{R_1} =_{df} \overline{R_1^1} \cup \overline{R_1^2}.$$

Finally, in accordance with Eq. (3),

$$C_{R_1} =_{df} R_1 \backslash \overline{R_1}$$

gives rise to a minimal complement $C = \{C_{R_1}\}$ of V with respect to D. Instance $C_{R_1}(d)$ is shown next.

C_{R_1}	Parent	Child
	Fred	Barney

3.2 Complements of PSJR Views

We next illustrate some complications in the computation of minimal complements in the presence of projections. Afterwards, we present our approach to the computation of complements for PSJR views, which is based on the equations given by Theorem 3.3, but which does not strive for minimality with respect to information content.

Example 3.6. Consider a single view $V =_{df} \pi_A(R)$ over $D = \{R\}$, where $attr(R) = key(R) = \{A, B\}$. As we have noticed above, a copy of D is a trivial complement for any set of views. Hence, $C = \{C_R\}$, where $C_R =_{df} R$, is a trivial

complement of V. (For simplicity, we use "V" instead of "$\{V\}$," i.e., we implicitly extend the notion of complement to single views.) However, this trivial complement is not the only relational complement possible and, in particular, not a minimal one (as implied by Example 22.3.8 of Abiteboul et al. [1995]).

Next, let t_0 be an arbitrary tuple over R. Then, $C' = \{C'_R\}$, where

$$C'_R =_{df} R \setminus [\{t_0\} \setminus (\pi_A(R \setminus \{t_0\}) \bowtie \{t_0\})],$$

is a complement of V with respect to D, as the following computation of R from V and C' shows:

$$R \approx C'_R \cup [(V \bowtie \{t_0\}) \setminus (\pi_A(C'_R) \bowtie \{t_0\})]$$

Moreover, we have $C' < C$, as $C'_R(\emptyset) = C'_R(\langle\{t_0\}\rangle) = \emptyset$, whereas $C_R(\emptyset) = \emptyset \neq \langle\{t_0\}\rangle = C_R(\langle\{t_0\}\rangle)$.

We remark that the following idea lies as the heart of the complement given in Example 3.6: Let t_0 be a fixed tuple over R. Then, we can construct an expression within which t_0 is stored (via a constant relation) and which "recognizes" instances r of R, where $t_0(A)$ occurs only once in column A (indeed, the subexpression $\{t_0\} \setminus (\pi_A(R \setminus \{t_0\}) \bowtie \{t_0\})$ of C'_R is nonempty if and only if $t_0(A)$ occurs only once in column A). For such relation instances we do not need to store t_0 in the complementary instance, as the A-value $t_0(A)$ occurs in the view instance of V and tuple t_0 is stored within the expression for C'_R.

3.2.1 *Why Minimality May Not Be An Issue.* Based on the last remark, we now observe that the complement of Example 3.6 is not minimal, as it can be further reduced by "recognizing" finite functions in the following sense: In the definition of C'_R and in the computation of R we can replace the one-tuple relation $\{t_0\}$ by any relation r_0 that satisfies the functional dependency $A \rightarrow B$. Clearly, the more tuples r_0 contains the smaller the resulting complement gets.

However, the above complement does not seem natural for the following reasons: First of all, the tuples that do not have to be stored in *instances* of the complementary view C'_R are still stored somewhere else, namely inside a constant relation hidden in the *expression* C'_R. This constant relation has to be stored for every instance of R, even for the empty instance. Next, the complement in the above example is designed to "watch out" for a certain tuple, which might never occur during the lifetime of the database. Therefore, it is questionable whether the reduced information content of the complement pays off with respect to the increased complexity of complementary views and view inverses. Finally, the above complement violates the property of *genericity*, which captures the data independence principle for database queries [Aho and Ullman 1979; Chandra and Harel 1980].

Loosely speaking, a view is generic if it treats data values essentially as uninterpreted objects (and, in particular, does *not* watch out for certain tuples). Formally, a view is *generic* if it commutes with all permutations of the database domain. Let A be an attribute. A mapping $p : dom(A) \rightarrow dom(A)$ is called a *permutation* (*of* $dom(A)$), if p is bijective. We extend the notion of permutation to database schemata in the natural way, and we say that p is a *permutation of* D if p gives rise to a permutation for each domain associated with an attribute

occurring in D. Moreover, given an instance d of D (or an instance $V(d)$ for some view V over D), the *result of p applied to d, $p(d)$,* (or to $V(d)$, $p(V(d))$) is the instance where for each domain $dom(A)$ associated with D the values of $dom(A)$ occurring in d (or in $V(d)$) are permuted according to the permutation for $dom(A)$ to which p gives rise.

Definition 3.7. Let V be a view over D. Then V is *generic* if we have $V(p(d)) = p(V(d))$ for all permutations p of D and instances d of D.

We believe that genericity is a desirable property for view complements, as potential reductions in the view size should not depend on the instance-specific occurrence of a finite number of constants. In the context of complements, we say that a *complement C is generic* if all views in C and additionally the view inverse defined by C are generic. (Note that genericity of the complement does not imply genericity of the view inverse: Reconsider Example 3.6 with exchanged roles of V and C, i.e., treat V as generic complement of view C'; clearly, any inverse must deal with the constant tuple t_0, i.e., cannot be generic.) It is straightforward to verify that the complements produced by Theorem 3.3 are generic if the underlying views V are.

However, restricting ourselves to generic complements in the current setting again yields complements that are smaller than the trivial one, but which are not useful in practice.

Example 3.8. Consider $C'' = \{C_B, C_{AB}\}$, where

$$C_B(r) =_{df} \begin{cases} \pi_B(r), & \text{if } r = \pi_B(r) \bowtie V(r) \\ \emptyset, & \text{otherwise} \end{cases}$$

and

$$C_{AB} =_{df} R \setminus (V \bowtie C_B).$$

Then C'' is a complement of V as $R \approx C_{AB} \cup (V \bowtie C_B)$. Moreover, it is easy to see that C'' is generic and that we have $C'' < C$. We point out that C'' is indeed a *relational* complement, as the definition of C_B is just a shorthand for

$$\pi_B(r) \bowtie [\{\langle\rangle\} \setminus \pi_\emptyset(r \setminus (\pi_B(r) \bowtie V(r)))],$$

where $\{\langle\rangle\}$ denotes the non-empty relation over the empty set of attributes (which is the neutral element with respect to join).

We note that the optimization concerning C'' is based on the computation of a lossless join (which degenerates to a Cartesian product) within base relation *instances*: Even if there is no lossless join constraint in general, some of the instances of D satisfy this constraint, which leads to the above reductions with respect to information content. Thus, although C'' is a generic complement, its reduced size basically results from the satisfaction of a constraint that may accidentally hold in particular instances. In practice, the exploitation of such "random" constraints will lead to marginal reductions in the size of complement *instances* only, while the costs involved in using and maintaining the complement (where each complementary view involves a Cartesian product) are prohibitively expensive.

Moreover, if a complement of $V =_{df} \pi_A(R)$ is generic, then the reductions illustrated in Example 3.6 can be applied to obtain smaller complements. Thus, we have:

LEMMA 3.9. *If a complement of view $V =_{df} \pi_A(R)$ over $D = \{R\}$, where $attr(R) = \{A, B\}$, is generic then it is not minimal with respect to $\leq_{ic}$.*

PROOF. Assume that C is a generic complement of V and consider the instance $r_0 = \{\langle a_0, b_0\rangle\}$ of R, where $a_0 \neq b_0$. (Notice that such an instance exists due to the assumption that domains contain at least two values.) As C is a generic complement, the B-value b_0 that is not visible in $V(r_0)$ must be visible in $C(r_0)$ to compute r_0 from $V(r_0)$ and $C(r_0)$, that is, we have $C(r_0) \neq \emptyset$.

In the spirit of Example 3.6, one can design a nongeneric complement C_0 to "watch out" for the instance $C(r_0)$ in such a way that $C_0(r_0)$ is empty:

$$C_0(r) =_{df} \begin{cases} \emptyset, & \text{if } C(r) = C(r_0) \text{ and } V(r) = V(r_0) \\ C(r), & \text{otherwise.} \end{cases}$$

Clearly, we have $C_0 <_{ic} C$. (Note that $C_0(\emptyset) = C_0(r_0) = \emptyset = C(\emptyset) \neq C(r_0)$.) Moreover, C_0 is a complement as we can compute C from V and C_0 (and then R from V and C, which concludes the proof):

$$C(r) = \begin{cases} C(r_0), & \text{if } C_0(r) = \emptyset \text{ and } V(r) = V(r_0) \\ C_0(r), & \text{otherwise.} \end{cases} \quad \square$$

Roughly speaking, Lemma 3.9 implies that minimal complements of $V =_{df} \pi_A(R)$ must be unnatural objects, which violate the property of genericity. Moreover, the above examples indicate that a complement of V that is smaller than the trivial one may not be useful in practice. Therefore, the question arises how such undesirable complements can be avoided. In this respect, we observe that the undesirable, "optimized" complements C' and C'' above are not super-key based complements according to Definition 2.7. In fact, this observation is not just coincidence:

LEMMA 3.10. *The trivial complement $C = \{C_R =_{df} R\}$ is minimal among super-key based complements for $V =_{df} \pi_A(R)$, that is, if C' is a super-key based complement for V, then we have $C \leq_{ic} C'$.*

PROOF. Assume for the sake of a contradiction that C' is a super-key based complement of V such that $C' <_{ic} C$. Let r_1 and r_2 be instances of R such that $C'(r_1) = C'(r_2)$ and $C(r_1) \neq C(r_2)$. As $C = \{C_R\}$ is just a copy of $D = \{R\}$, we have $C_R(r_1) = r_1 \neq r_2 = C_R(r_2)$. As C' is a super-key based complement and as the key of R is the trivial one, C' contains a single view, say C'_R. Clearly, C'_R is not an empty view (as, e.g., the instances $\{(a_0, b_0)\}$ and $\{(a_0, b_1)\}$ of R, which are not distinguished by V, must be distinguished by any complement). Hence, we have $C'_R \approx R \backslash E$, where E is a UPSJR expression over V. In particular, we obtain $C'_R \subseteq R$. Hence, we have $C'_R(r_1) = C'_R(r_2) \subseteq r_1$ and $C'_R(r_1) = C'_R(r_2) \subseteq r_2$. Due to $r_1 \neq r_2$, at least one of the previous inclusions must be strict, say $C'_R(r_1) = C'_R(r_2) \subsetneq r_1$.

Without loss of generality assume that there is $t_0 \in r_1 \backslash C'_R(r_1)$. Consider an instance r_3 of R such that $V(r_3) = V(r_1)$, $t_0 \in r_3$, and $t_1 \in r_3$, where $t_1 \neq t_0$ and

$t_0(A) = t_1(A)$. (Notice that such an instance exists due to the assumption that domains contain at least two values.) Moreover, consider $r_4 = r_3 \setminus \{t_0\}$. Then, we have $V(r_1) = V(r_3) = V(r_4)$. Hence, E yields the same result r_5 for these three view instances. In particular, as $t_0 \notin C'_R(r_1)$ we have $t_0 \in r_5$. Consequently, we deduce $t_0 \notin C'_R(r_3) = C'_R(r_4)$. To summarize, r_3 and r_4 are relation instances such that

—$r_3 \neq r_4$,
—$V(r_3) = V(r_4)$,
—$C'_R(r_3) = C'_R(r_4)$,

which contradicts Proposition 2.6 and concludes the proof, showing that C is minimal among super-key based complements. □

As C is the trivial complement (which implies $C' \leq_{ic} C$ for any complement C'), Lemma 3.10 can be interpreted as follows: Among the super-key based complements, there is only one complement for $V = \pi_A(R)$ (up to equivalence), namely the trivial one.

3.2.2 *Why Super-Key Based Complements Are Good for Projections.* In light of the above results, we argue that it does not always make sense to look for minimal complements, in particular if projections are involved. Instead, we pursue a more pragmatic approach, which is based on the computation of super-key based complements and takes advantage of key constraints in the computation of complements for views involving projections. The key idea in the following is that projections that preserve keys of relation schemata may permit the computation of lossless joins, an observation that has been made by Honeyman [1980] in an entirely different context and that has lead to the notion of an *extension join*.

Example 3.11. Let $D = \{R\}$, where $attr(R) = \{A, B, C\}$ and $key(R) = \{A\}$. Let $V_1 =_{df} \pi_{AB}(R)$ and $V = \{V_1\}$. Since V_1 preserves the key of R, we have a lossless join $R \approx V_1 \bowtie \pi_{AC}(R)$. Hence, $C = \{C_R^{AC} =_{df} \pi_{AC}(R)\}$ is a complement of V. Moreover, it is easy to see that C is strictly smaller than the trivial complement and that C is super-key based.

Let D be a set of relation schemata, and let V be a set of PSJR views over D. Our goal is to compute complements such as the one of Example 3.11 based on the equations provided by Theorem 3.3, although the latter does not take projections into account. The roadmap is as follows: We first perform a schema transformation on D which preserves the information content of D and builds a "virtual database" D^v, where each relation schema either has a trivial key or contains exactly one non-key attribute in addition to the key. (We call this a virtual database, because we only make use of the *expressions* involved in this database, but we never store any of its parts.) Afterwards, we rewrite V over D^v and obtain a set V^v of PSJR views over D^v. Then, by applying a variant of Theorem 3.3 to D^v, we obtain a super-key based complement C^v of V with respect to D^v (cf. Theorem 3.17). It turns out that C^v is also a complement of V with respect to D (cf. Corollary 3.18) and that

C^v gives rise to a super-key based complement C of V with respect to D (cf. Corollary 3.19).

Definition 3.12. Let D be a database schema. Let $D_t \subseteq D$ be a set of views that contains copies of those relation schemata that have less than two non-key attributes, that is,

$$D_t = \left\{ R^{attr(R)} =_{df} R \mid R \in D \ \wedge \ |attr(R)| - |key(R)| < 2 \right\}.$$

Let $D_n = D \backslash D_t$, and let D_n^v be the set of views that is obtained by creating $|attr(R)| - |key(R)|$ projection views for $R \in D_n$, where each projection contains the key of R and a single non-key attribute, that is,

$$D_n^v = \{R^X =_{df} \pi_X(R) \mid R \in D_n \ \wedge \ key(R) \subsetneqq X \subseteq attr(R) \ \wedge \ |X| - |key(R)| = 1\}.$$

Let $D^v = D_t \cup D_n^v$. We call the set of views D^v the *virtual database for* D. Moreover, for $R \in D$ we refer to the expressions in D^v that involve R as the *virtual relations for* R.

We first verify that the information content of a set of relation schemata is preserved in its virtual database.

LEMMA 3.13. *Let D be a database schema, and let D^v be the virtual database for D. Then $D \leq_{ic} D^v$.*

PROOF. We show that D can be computed from D^v, from which the claim follows. Let $R \in D$. If R has a less than two non-key attributes then a copy of R is contained in D^v. Hence, R can be computed from D^v. Otherwise, we have $n = |attr(R)| - |key(R)| \geq 2$, and D^v contains n projections of the form $\pi_{X_i}(R)$, where $key(R) \subseteq X_i$, $1 \leq i \leq n$. Therefore, we have a lossless join $R \approx \pi_{X_1}(R) \bowtie \cdots \bowtie \pi_{X_n}(R)$, which concludes the proof. □

Example 3.14. For $D = \{R(\underline{A}, B, C)\}$ as in Example 3.11, we obtain $D^v = \{R^{AB} =_{df} \pi_{AB}(R), R^{AC} =_{df} \pi_{AC}(R)\}$. Moreover, R^{AB} and R^{AC} are the virtual relations for R. As the key of R is contained in each projection, we have a lossless join $R \approx R^{AB} \bowtie R^{AC}$.

Next, we rewrite sets of views in terms of virtual databases. As we have explained in Section 2.1, a view can be regarded as a relation schema, and a view instance is just a relation instance. Based on this point of view, in the following definition, the set of views defining the virtual database is treated as a set of relation schemata.

Definition 3.15. Let D be a database schema with virtual database D^v, and let V be a set of PSJR views over D. Then the *rewriting of V over D^v*, denoted by V^v, is a set of PSJR views over D^v that contains one view per view in V, where

(1) each reference to a relation schema R is replaced by a join of the virtual relations for R and
(2) a renaming function, which is applied to R in the original view, is pushed down to each virtual relation for R.

Example 3.16. Let V_0 be a PSJR view, that is,

$$V_0 =_{df} \pi_{X_i}(\sigma_{\phi_i}(\rho_{f_{i,i_1}}(R_{i_1}) \bowtie \cdots \bowtie \rho_{f_{i,i_k}}(R_{i_k}))).$$

Assume that there are n_{i_j} virtual relations for R_{i_j}. Then, in a first step, we obtain:

$$\pi_{X_i}\left(\sigma_{\phi_i}\left(\rho_{f_{i,i_1}}\left(R_{i_1}^1 \bowtie \cdots \bowtie R_{i_1}^{n_{i_1}}\right) \bowtie \cdots \bowtie \rho_{f_{i,i_k}}\left(R_{i_k}^1 \bowtie \cdots \bowtie R_{i_k}^{n_{i_k}}\right)\right)\right)$$

Afterwards, by pushing down renaming functions, we obtain the rewriting of V_0:

$$\pi_{X_i}\left(\sigma_{\phi_i}\left(\rho_{f_{i,i_1}}\left(R_{i_1}^1\right) \bowtie \cdots \bowtie \rho_{f_{i,i_1}}\left(R_{i_1}^{n_{i_1}}\right) \bowtie \cdots \bowtie \rho_{f_{i,i_k}}\left(R_{i_k}^1\right) \bowtie \cdots \bowtie \rho_{f_{i,i_k}}\left(R_{i_k}^{n_{i_k}}\right)\right)\right).$$

Given a set of PSJR views V over D and its virtual database D^v, the following Theorem 3.17 formalizes how a complement of V with respect to D^v can be computed using the rewriting of V over D^v.

THEOREM 3.17. *Let D be a database schema with virtual database $D^v = \{R_1, \ldots, R_n\}$, and let V be a set of PSJR views over D. Let V^v be the rewriting of V over D^v, where each $V_i^v \in V^v$, which is the rewriting of $V_i \in V$, has the following form:*

$$V_i^v =_{df} \pi_{X_i}\left(\sigma_{\phi_i}\left(\rho_{f_{i,i_1}}(R_{i_1}) \bowtie \cdots \bowtie \rho_{f_{i,i_k}}(R_{i_k})\right)\right)$$

For each $R_j \in D^v$ and each $V_i \in V$ let view $\overline{R_j^i}$ be defined as follows, where $attr(\overline{R_j^i}) = attr(R_j)$:

$$\overline{R_j^i} =_{df} \begin{cases} \emptyset & \text{if } R_j \text{ does not occur in } V_i^v, \\ \bigcup\limits_{\substack{R_{i_l} = R_j \\ f_{i,i_l}(attr(R_j)) \subseteq X_i}} \rho_{f_{i,i_l}^{-1}}(\pi_{f_{i,i_l}(attr(R_j))}(V_i)) & \text{otherwise.} \end{cases} \quad (9)$$

For $R_j \in D^v$ let view $\overline{R_j}$ be defined as follows:

$$\overline{R_j} =_{df} \bigcup_{V_i \in V} \overline{R_j^i}. \quad (10)$$

Then the set of views $C^v = \{C_{R_1}, \ldots, C_{R_n}\}$, where

$$C_{R_j} =_{df} R_j \backslash \overline{R_j}, \qquad 1 \leq j \leq n, \quad (11)$$

is a super-key based complement of V with respect to D^v. The view inverse is defined as follows for $R_j \in D^v$:

$$R_j \approx C_{R_j} \cup \overline{R_j}. \quad (12)$$

PROOF. To show that C^v is a complement of V, we first emphasize that the expressions given by Eq. (9) are *not* defined in terms of rewritten views in V^v but in terms of the original views V. Indeed, given a view $V_i \in V$ and a virtual relation R_j the rewritten view V_i^v is only used to determine whether $X_i (= attr(V_i) = attr(V_i^v))$ contains all of R_j's attributes (possibly in renamed form). If X_i does contain all of R_j's attributes then the expression $\rho_{f_{i,i_l}^{-1}}(\pi_{f_{i,i_l}(attr(R_j))}(V_i))$ is used to extract that portion of the instance of R_j that is preserved in an instance

of V_i, just as the same expression is used in Theorem 3.3. Hence, exactly as in the proof of Theorem 3.3, Eq. (9) defines an expression $\overline{R_j^i}$ such that $\overline{R_j^i} \subseteq R_j$, which shows that C^v is a complement of V with respect to D^v. Finally, C^v is super-key based by construction. □

The reader is encouraged to compare the expressions given by Theorem 3.17 with those given by Theorem 3.3 above for SJR views, keeping in mind that base relations in Theorem 3.3 correspond to virtual relations in Theorem 3.17. It turns out that only minor modifications concerning expression $\overline{R_j^i}$ are necessary to deal with PSJR views based on the expressions given by Theorem 3.3: While in the case of an SJR view V_i either all attributes of base relation R_j occur in the view (possibly in renamed form) or none of them, in case of a PSJR view some attributes may be projected away. Thus, in order to compute a "small" complementary views, the key issue is to identify those views that preserve a "useful" sets of attributes. Based on the intuition provided by Examples 3.11 and 3.14 we make use of those views that preserve one or more virtual relations, that is, that preserve super-keys of base relations. More precisely, expression $\overline{R_j^i}$ makes use of those *original views V_i in V* that preserve all the attributes of one or more virtual relations R_j by a syntactic check involving the *rewriting V_i^v of V_i over D^v*.

COROLLARY 3.18. *Let D, V, D^v, V^v, and C^v be as in Theorem* 3.17. *Then C^v is a complement of V with respect to D.*

PROOF. By Theorem 3.17, C^v is a complement of V with respect to D^v. As the proof of Lemma 3.13 shows how D can be computed relationally from D^v, C^v is also a complement of V with respect to D. □

COROLLARY 3.19. *Let D, V, D^v, V^v, and C^v be as in Theorem* 3.17. *Let C be the set of views over D that is obtained from C^v by replacing each reference to a virtual relation R' with the view definition of R', that is,*

$$C = \{C_R^X =_{df} \pi_X(R)\backslash\overline{R'} \mid (\exists C_{R'} =_{df} R'\backslash\overline{R'} \in C^v) \\ R' \text{ is a virtual relation for } R \wedge attr(R') = X\}.$$

Then C is a super-key based complement of V with respect to D. Moreover, if $R_j^{X_1}, \ldots, R_j^{X_n}$ are the virtual relations for $R_j \in D$, where $R_j^{X_i} \approx C_{R_j^{X_i}} \cup \overline{R_j^{X_i}}$ is the view inverse for $R_j^{X_i}$ given by Equivalence (12)*, then the view inverse for R_j is defined as follows:*

$$R_j \approx \left(C_{R_j^{X_1}} \cup \overline{R_j^{X_1}}\right) \bowtie \cdots \bowtie \left(C_{R_j^{X_n}} \cup \overline{R_j^{X_n}}\right). \tag{13}$$

PROOF. We recall that the virtual relations for relation schema R are just projections over R. Hence, the information content of C agrees with the information content of C^v. Consequently, as C^v is a complement of V by Corollary 3.18, C is also a complement of V. Furthermore, it is easy to see that Equivalence (13) is correct (as all joins are key-based). Finally, C is super-key based by construction. □

We emphasize that all join operations involved in the view inverses given by Equivalence (13) are extension joins and hence performed along keys; consequently, they can be computed using efficient algorithms [Honeyman 1980].

Example 3.20. Consider $D = \{R(\underline{A}, B, C, D, E)\}$ and views $V = \{V_1, V_2\}$ over D, where $V_1 =_{df} \pi_{ABC}(\sigma_\phi(R))$ and $V_2 =_{df} \pi_{AB}(R)$. Then, the virtual database D^v is given by the following views over D:

—$R^{AB} =_{df} \pi_{AB}(R)$,
—$R^{AC} =_{df} \pi_{AC}(R)$,
—$R^{AD} =_{df} \pi_{AD}(R)$,
—$R^{AE} =_{df} \pi_{AE}(R)$.

The rewriting V^v of V over D^v is given by the following views:

—$V_1^v =_{df} \pi_{ABC}(\sigma_\phi(R^{AB} \bowtie R^{AC} \bowtie R^{AD} \bowtie R^{AE}))$
—$V_2^v =_{df} \pi_{AB}(R^{AB} \bowtie R^{AC} \bowtie R^{AD} \bowtie R^{AE})$

In this setting, Eq. (11) of Theorem 3.17 produces the complementary views

$$\begin{aligned} C_{R^{AB}} &=_{df} R^{AB} \setminus (\pi_{AB}(V_1) \cup \pi_{AB}(V_2)), \\ C_{R^{AC}} &=_{df} R^{AC} \setminus \pi_{AC}(V_1), \\ C_{R^{AD}} &=_{df} R^{AD}, \\ C_{R^{AE}} &=_{df} R^{AE}, \end{aligned}$$

which—according to Equivalence (12)—allow to compute the virtual relations using the following inverses:

$$\begin{aligned} R^{AB} &\approx C_{R^{AB}} \cup (\pi_{AB}(V_1) \cup \pi_{AB}(V_2)), \\ R^{AC} &\approx C_{R^{AC}} \cup \pi_{AC}(V_1), \\ R^{AD} &\approx C_{R^{AD}}, \\ R^{AE} &\approx C_{R^{AE}}. \end{aligned}$$

Next Corollary 3.18 states that $C^v = \{C_{R^{AB}}, C_{R^{AC}}, C_{R^{AD}}, C_{R^{AE}}\}$, which is obviously a complement of V with respect to D^v, is a complement of V with respect to D. Indeed, Corollary 3.19 produces the super-key based complement $C = \{C_R^{AB}, C_R^{AC}, C_R^{AD}, C_R^{AE}\}$ of V with respect to D, where

$$\begin{aligned} C_R^{AB} &=_{df} \pi_{AB}(R) \setminus (\pi_{AB}(V_1) \cup \pi_{AB}(V_2)), \\ C_R^{AC} &=_{df} \pi_{AC}(R) \setminus \pi_{AC}(V_1), \\ C_R^{AD} &=_{df} \pi_{AD}(R), \\ C_R^{AE} &=_{df} \pi_{AE}(R). \end{aligned}$$

Moreover, according to Equivalence (13) the view inverse for R is given by:

$$R \approx \left(C_R^{AB} \cup (\pi_{AB}(V_1) \cup \pi_{AB}(V_2))\right) \bowtie \left(C_R^{AC} \cup \pi_{AC}(V_1)\right) \bowtie C_R^{AD} \bowtie C_R^{AE}$$

We observe from Example 3.20 that Theorem 3.17 and Corollary 3.19 may eventually generate a large number of complementary views for a single relation schema, where each view has a "small" schema. To be precise, for every

relation schema R we generate $\min\{1, |attr(R)| - |key(R)|\}$ complementary views whose schemata contain $|key(R)| + 1$ attributes each. Clearly, the view inverse for R then involves $\min\{0, |attr(R)| - |key(R)| - 1\}$ joins. Furthermore, based on the expressions of Example 3.20 it requires little thought to see that some of the joins involved can be avoided.

Example 3.21. In continuation of the previous example, consider the views

$$\begin{aligned} C_R^{AB} &=_{df} \pi_{AB}(R)\backslash(\pi_{AB}(V_1) \cup \pi_{AB}(V_2)), \\ C_R^{AC} &=_{df} \pi_{AC}(R)\backslash\pi_{AC}(V_1), \text{ and} \\ C_R^{ADE} &=_{df} \pi_{ADE}(R) \end{aligned}$$

that form a complement C' of V as a view inverse for R is given by

$$R \approx \left(C_R^{AB} \cup (\pi_{AB}(V_1) \cup \pi_{AB}(V_2))\right) \bowtie \left(C_R^{AC} \cup \pi_{AC}(V_1)\right) \bowtie C_R^{ADE},$$

which saves one join in comparison to the above inverse for C.

Notice that both complements have the same information content, while query processing costs associated with the inverse of C' are reduced.

To generalize the idea for the improvement in the previous example, we recall from Definition 2.7 that, given a complement C and relation R, we use $C[R]$ to denote those complementary views in C whose definition is equivalent to the form $\pi_X(R)\backslash E$ for $X \subseteq attr(R)$ and an arbitrary expression E. Now, in order to reduce the number of joins involved in view inverses given by Corollary 3.19, we have to identify those complementary views in $C[R] \subseteq C$ (where C is the complement for a set V of PSJR views given by Corollary 3.19) that can be "combined" into a single one. For this purpose, we use $\mathcal{P}(C[R])$ to denote the coarsest partitioning $\{C[R]_1, \ldots, C[R]_{k(R)}\}$ of $C[R]$ such that if one view in $C[R]_i$ accesses a view $V_j \in V$ within a subexpression $\rho_{f_{j_i}^{-1}}(\pi_{f_{j_i}(X)}(V_j))$ then all views in $C[R]_i$ access V_j within a subexpression $\rho_{f_{j_i}^{-1}}(\pi_{f_{j_i}(X')}(V_j))$ (where the same renaming function is used but the set of projected attributes may differ), $1 \leq i \leq k(R)$. The complementary views in each set of the partitioning $\mathcal{P}(C[R])$ will then be combined into a single complementary view.

THEOREM 3.22. *Let D, V, and C be as in Corollary* 3.19. *Given $R \in D$, let $\mathcal{P}(C[R]) = \{C[R]_1, \ldots, C[R]_{k(R)}\}$.*

For $i \in [1, k(R)]$, let $C[R]_i = \{C_R^{X_{i_1}}, \ldots, C_R^{X_{i_{n_i}}}\}$, where

$$C_R^{X_{i_l}} =_{df} \pi_{X_{i_l}}(R)\backslash\left(\rho_{f_{j_1}^{-1}}\left(\pi_{f_{j_1}(X_{i_l})}(V_{j_1})\right) \cup \cdots \cup \rho_{f_{j_{c_i}}^{-1}}\left(\pi_{f_{j_{c_i}}(X_{i_l})}\left(V_{j_{c_i}}\right)\right)\right),$$

for $c_i \geq 1$ and $1 \leq l \leq n_i$, and let

$$X_{R,i} = \bigcup_{l=1}^{n_i} X_{i_l}.$$

For $i \in [1, k(R)]$ define view $\overline{R^{X_{R,i}}}$ as follows:

$$\overline{R^{X_{R,i}}} =_{df} \left(\rho_{f_{j_1}^{-1}}\left(\pi_{f_{j_1}(X_{R,i})}(V_{j_1})\right) \cup \cdots \cup \rho_{f_{j_{c_i}}^{-1}}\left(\pi_{f_{j_{c_i}}(X_{R,i})}\left(V_{j_{c_i}}\right)\right)\right) \tag{14}$$

For $i \in [1, k(R)]$, define view $C_R^{X_{R,i}}$ as follows:

$$C_R^{X_{R,i}} =_{df} \pi_{X_{R,i}}(R) \backslash \overline{R^{X_{R,i}}} \tag{15}$$

Then, the set of views over D defined by Eq. (15) for all $R \in D$ and $i \in [1, k(R)]$ is a super-key based complement of V with respect to D. The view inverse is defined as follows for $R \in D$:

$$R \approx \left(C_R^{X_{R,1}} \cup \overline{R^{X_{R,1}}}\right) \bowtie \cdots \bowtie \left(C_R^{X_{R,k(R)}} \cup \overline{R^{X_{R,k(R)}}}\right) \tag{16}$$

To prepare the proof of the above theorem, we first state and prove a technical lemma.

LEMMA 3.23. *Let D be a set of relation schemata, let $R \in D$, and let V_1 and V_2 be PSJR views over D such that $\rho_f(R)$ occurs as subexpression in V_1 and V_2. Moreover, let key$(R) \subseteq X_i \subseteq attr(R)$, $i = 1, 2$. If $f(X_i) \subseteq attr(V_j)$, $i, j = 1, 2$, then*

$$\begin{aligned} &\rho_{f^{-1}}\left(\pi_{f(X_1 \cup X_2)}(V_1)\right) \cup \rho_{f^{-1}}\left(\pi_{f(X_1 \cup X_2)}(V_2)\right) \\ &\approx \left(\rho_{f^{-1}}\left(\pi_{f(X_1)}(V_1)\right) \cup \rho_{f^{-1}}\left(\pi_{f(X_1)}(V_2)\right)\right) \\ &\quad \bowtie \left(\rho_{f^{-1}}\left(\pi_{f(X_2)}(V_1)\right) \cup \rho_{f^{-1}}\left(\pi_{f(X_2)}(V_2)\right)\right). \end{aligned} \tag{17}$$

PROOF. We first show some basic properties of the subexpressions involved in Equivalence (17). Afterwards, we use these properties to show both inclusions of the equivalence. To simplify notation, let

$$E_X(V_i) =_{df} \rho_{f^{-1}}\left(\pi_{f(X)}(V_i)\right),$$

where $f(X) \subseteq attr(V_i)$ and $i = 1, 2$, that is, Equivalence (17) becomes:

$$\begin{aligned} &E_{X_1 \cup X_2}(V_1) \cup E_{X_1 \cup X_2}(V_2) \\ &\approx (E_{X_1}(V_1) \cup E_{X_1}(V_2)) \bowtie (E_{X_2}(V_1) \cup E_{X_2}(V_2)) \end{aligned} \tag{18}$$

As in the proof of Theorem 3.3, we have $E_{X_i}(V_j) \subseteq \pi_{X_i}(R)$, $i, j = 1, 2$. As X_1 and X_2 contain the key of R we can deduce:

$$E_{X_1 \cup X_2}(V_i) \approx E_{X_1}(V_i) \bowtie E_{X_2}(V_i), \quad i = 1, 2 \tag{19}$$

Using Equivalence (19), we have the following equivalence concerning the left-hand side of Equivalence (18):

$$E_{X_1 \cup X_2}(V_1) \cup E_{X_1 \cup X_2}(V_2) \approx (E_{X_1}(V_1) \bowtie E_{X_2}(V_1)) \cup (E_{X_1}(V_2) \bowtie E_{X_2}(V_2)) \tag{20}$$

As join distributes over union, for the right hand side of Equivalence (18) we have:

$$\begin{aligned} &(E_{X_1}(V_1) \cup E_{X_1}(V_2)) \bowtie (E_{X_2}(V_1) \cup E_{X_2}(V_2)) \\ &\approx (E_{X_1}(V_1) \bowtie E_{X_2}(V_1)) \cup (E_{X_1}(V_1) \bowtie E_{X_2}(V_2)) \\ &\quad \cup (E_{X_1}(V_2) \bowtie E_{X_2}(V_1)) \cup (E_{X_1}(V_2) \bowtie E_{X_2}(V_2)) \end{aligned} \tag{21}$$

Now, from Equivalences (20) and (21), we obtain the first inclusion:

$$E_{X_1 \cup X_2}(V_1) \cup E_{X_1 \cup X_2}(V_2) \subseteq (E_{X_1}(V_1) \cup E_{X_1}(V_2)) \bowtie (E_{X_2}(V_1) \cup E_{X_2}(V_2))$$

Concerning the other inclusion, based on Equivalence (20), we have to show:

$$(E_{X_1}(V_1) \bowtie E_{X_2}(V_2)) \cup (E_{X_1}(V_2) \bowtie E_{X_2}(V_1)) \subseteq E_{X_1 \cup X_2}(V_1) \cup E_{X_1 \cup X_2}(V_2) \quad (22)$$

To this end, consider an instance d of D and a tuple t in the instance of the left expression in Inclusion (22), that is, in

$$(E_{X_1}(V_i) \bowtie E_{X_2}(V_j))(d),$$

where $i, j = 1, 2$ such that $i \neq j$. Without loss of generality let $i = 1$, $j = 2$ (a symmetric argument holds for the case $i = 2$, $j = 1$). We show that t is contained in the instance

$$(E_{X_1 \cup X_2}(V_1))(d) = (E_{X_1}(V_1) \bowtie E_{X_2}(V_1))(d),$$

(note that the choice of V_1 in this equation is arbitrary; we could equally well use V_2 here) from which Inclusion (22) follows. From $E_{X_i}(V_j) \subseteq \pi_{X_i}(R)$, $i, j = 1, 2$, and $key(R) \subseteq X_1 \cap X_2$ we deduce $t \in \pi_{X_1 \cup X_2}(R(d))$. As $key(R) \subseteq X_1$ and $X_1 \cup X_2 = attr(E_{X_1 \cup X_2}(V_1))$, we have

$$t \in (E_{X_1}(V_1) \bowtie E_{X_2}(V_1))(d),$$

which concludes the proof. □

PROOF OF THEOREM 3.22. We have to show that Equivalence (16) is correct, which then implies that the set of views defined by Eq. (15) is a complement of V with respect to D. Moreover, by construction this complement is super-key based.

We note that Equivalence (16) is related to Equivalence (12) as follows: By definition, each subexpression

$$C_R^{X_{R,i}} \cup \overline{R^{X_{R,i}}}$$

in Equivalence (16) replaces a join of $n_i \geq 1$ subexpressions of the form

$$\left(C_R^{X_{i_1}} \cup \overline{R^{X_{i_1}}}\right), \ldots, \left(C_R^{X_{i_{n_i}}} \cup \overline{R^{X_{i_{n_i}}}}\right)$$

in Equivalence (12) such that

$$C[R]_i = \left\{C_R^{X_{i_1}}, \ldots, C_R^{X_{i_{n_i}}}\right\}, \text{ and } X_{R,i} = \bigcup_{l=1}^{n_i} X_{i_l}.$$

Our goal is to establish that this replacement of expressions is equivalence-preserving, that is,

$$C_R^{X_{R,i}} \cup \overline{R^{X_{R,i}}} \approx \left(C_R^{X_{i_1}} \cup \overline{R^{X_{i_1}}}\right) \bowtie \cdots \bowtie \left(C_R^{X_{i_{n_i}}} \cup \overline{R^{X_{i_{n_i}}}}\right), \quad (23)$$

which then allows to deduce correctness of Equivalence (16) from the correctness of Equivalence (12). For this purpose, we verify

$$\overline{R^{X_{R,i}}} \approx \overline{R^{X_{i_1}}} \bowtie \cdots \bowtie \overline{R^{X_{i_{n_i}}}}, \quad (24)$$

which implies

$$\overline{R^{X_{R,i}}} \subseteq \pi_{X_{R,i}}(R). \quad (25)$$

(Indeed, as in the proof of Theorem 3.17 we have $\overline{R^{X_{i_l}}} \subseteq \pi_{X_{i_l}}(R), l \in [1, n_i]$. Thus, Equivalence (24) and $\pi_{X_{R,i}}(R) \approx \pi_{X_{i_1}}(R) \bowtie \cdots \bowtie \pi_{X_{i_{n_i}}}(R)$ imply $\overline{R^{X_{R,i}}} \subseteq \pi_{X_{R,i}}(R)$.) Observing that

$$\begin{aligned}\pi_{X_{R,i}}(R) &\approx \pi_{X_{i_1}}(R) \bowtie \cdots \bowtie \pi_{X_{i_{n_i}}}(R) \\ &\approx \left(C_R^{X_{i_1}} \cup \overline{R^{X_{i_1}}}\right) \bowtie \cdots \bowtie \left(C_R^{X_{i_{n_i}}} \cup \overline{R^{X_{i_{n_i}}}}\right)\end{aligned}$$

and

$$C_R^{X_{R,i}} =_{df} \pi_{X_{R,i}}(R) \backslash \overline{R^{X_{R,i}}},$$

we then have Equivalence (23) as follows, where Inclusion (25) assures that difference followed by union is idempotent:

$$\begin{aligned}C_R^{X_{R,i}} \cup \overline{R^{X_{R,i}}} &\approx (\pi_{X_{R,i}}(R) \backslash \overline{R^{X_{R,i}}}) \cup \overline{R^{X_{R,i}}} \\ &\approx \pi_{X_{R,i}}(R) \\ &\approx \left(C_R^{X_{i_1}} \cup \overline{R^{X_{i_1}}}\right) \bowtie \cdots \bowtie \left(C_R^{X_{i_{n_i}}} \cup \overline{R^{X_{i_{n_i}}}}\right)\end{aligned}$$

Hence, it remains to verify Equivalence (24). Using the definitions for view names in Equivalence (24) we have to verify:

$$\begin{aligned}&\rho_{f_{j_1}^{-1}}\left(\pi_{f_{j_1}(X_{R,i})}(V_{j_1})\right) \cup \cdots \cup \rho_{f_{j_{c_i}}^{-1}}\left(\pi_{f_{j_{c_i}}(X_{R,i})}\left(V_{j_{c_i}}\right)\right) \\ &\qquad \approx \bowtie_{l=1}^{n_i}\left(\rho_{f_{j_1}^{-1}}\left(\pi_{f_{j_1}(X_{i_l})}(V_{j_1})\right) \cup \cdots \cup \rho_{f_{j_{c_i}}^{-1}}\left(\pi_{f_{j_{c_i}}(X_{i_l})}\left(V_{j_{c_i}}\right)\right)\right)\end{aligned}$$

Recalling $X_{R,i} = \bigcup_{l=1}^{n_i} X_{i_l}$, where $key(R) \subseteq X_{i_l}$, the claim now follows from Lemma 3.23, which concludes the proof. □

To end the discussion of complements for PSJR views, we exhibit two examples concerning Theorem 3.22, showing that some joins are indeed avoided, and we note that, for this case, no previous results were known.

Example 3.24. For Example 3.20, it is easily verified that Theorem 3.22 produces complement C' with the associated inverse that saves a join.

Example 3.25. Consider relation schemata $D = \{R_1, R_2\}$ and views $V = \{V_1, V_2, V_3\}$ over D, where

—$R_1(\underline{A}, B, C_1, \ldots, C_n)$,
—$R_2(\underline{D}, A, E, F_1, \ldots, F_m)$,
—$V_1 =_{df} \pi_{ABDE}(\sigma_{\phi_1}(R_1 \bowtie R_2))$,
—$V_2 =_{df} \pi_{ADE}(\sigma_{\phi_2}(R_2))$,
—$V_3 =_{df} \pi_{BC_1EG}(\sigma_{\phi_3}(R_1 \bowtie \rho_f(R_2)))$, where f renames D into C_1 and A into G.

Then Corollary 3.19 leads to the following complementary views, where the view inverses for R_1 and R_2 involve n and $m+1$ joins, respectively:

$$\begin{aligned}
C_{R_1}^{AB} &=_{df} \pi_{AB}(R_1)\backslash\pi_{AB}(V_1)\\
C_{R_1}^{AC_1} &=_{df} \pi_{AC_1}(R_1)\\
&\vdots\\
C_{R_1}^{AC_n} &=_{df} \pi_{AC_n}(R_1)\\
C_{R_2}^{DA} &=_{df} \pi_{DA}(R_2)\backslash(\pi_{DA}(V_1)\cup\pi_{DA}(V_2)\cup\rho_{f^{-1}}(\pi_{C_1G}(V_3)))\\
C_{R_2}^{DE} &=_{df} \pi_{DE}(R_2)\backslash(\pi_{DE}(V_1)\cup\pi_{DE}(V_2)\cup\rho_{f^{-1}}(\pi_{C_1E}(V_3)))\\
C_{R_2}^{DF_1} &=_{df} \pi_{DF_1}(R_2)\\
&\vdots\\
C_{R_2}^{DF_m} &=_{df} \pi_{DF_m}(R_2).
\end{aligned}$$

Let $C = \{C_{R_1}^{AB}, C_{R_1}^{AC_1}, \ldots, C_{R_1}^{AC_n}, C_{R_2}^{DA}, C_{R_2}^{DE}, C_{R_2}^{DF_1}, \ldots, C_{R_2}^{DF_m}\}$. Using the notation of Theorem 3.22, we have

—$C[R_1] = \{C_{R_1}^{AB}, C_{R_1}^{AC_1}, \ldots, C_{R_1}^{AC_n}\}$ and
—$C[R_2] = \{C_{R_2}^{DA}, C_{R_2}^{DE}, C_{R_2}^{DF_1}, \ldots, C_{R_2}^{DF_m}\}$.

The coarsest partitionings for both sets are given by:

$$\begin{aligned}
C[R_1]_1 &= \{C_{R_1}^{AB}\}\\
C[R_1]_2 &= \{C_{R_1}^{AC_1}, \ldots, C_{R_1}^{AC_n}\}\\
C[R_2]_1 &= \{C_{R_2}^{DA}, C_{R_2}^{DE}\}\\
C[R_2]_2 &= \{C_{R_2}^{DF_1}, \ldots, C_{R_2}^{DF_m}\}
\end{aligned}$$

Thus, Theorem 3.22 leads to the following complementary views, where the view inverses for R_1 and R_2 involve only a single join each:

$$\begin{aligned}
C_{R_1}^{AB} &=_{df} \pi_{AB}(R_1)\backslash\pi_{AB}(V_1)\\
C_{R_1}^{AC_1\ldots C_n} &=_{df} \pi_{AC_1\ldots C_n}(R_1)\\
C_{R_2}^{DAE} &=_{df} \pi_{DAE}(R_2)\backslash(\pi_{DAE}(V_1)\cup\pi_{DAE}(V_2)\cup\rho_{f^{-1}}(\pi_{C_1EG}(V_3)))\\
C_{R_2}^{DF_1\ldots F_m} &=_{df} \pi_{DF_1\ldots F_m}(R_2)
\end{aligned}$$

3.3 Complements of UPSJR Views

Our next goal is to extend the computation of complements to views involving union. Consider a view $V =_{df} R_1 \cup R_2$ and assume that there are selection conditions ϕ_1 and ϕ_2 such that $\sigma_{\phi_i}(V) \approx R_i$, $i = 1, 2$. In such a situation, we clearly do not need to store any complementary information because both relations can be computed using selection. Indeed, even if such selection conditions do not exist in the first place, we can enforce them by modifying the view definition to

include a source identifier: By changing the definition of V to

$$(R_1 \bowtie \{(1)\}) \cup (R_2 \bowtie \{(2)\}), \tag{26}$$

where we assume $\{(1)\}$ and $\{(2)\}$ to be (constant) relations over a new schema with a single attribute, say *Source*, there is no need for a complement, as the relations can be computed from V as follows:

$$R_1 \approx \pi_{attr(R_1)}(\sigma_{Source=1}(V)),\ R_2 \approx \pi_{attr(R_2)}(\sigma_{Source=2}(V)).$$

We call union views of the form given by Expression (26) *source-preserving union views*.

The following theorem formalizes the computation of complements for source-preserving union views.

THEOREM 3.26. *Let $U = \{U_1, \ldots, U_m\}$ be a set of UPSJR views over $D = \{R_1, \ldots, R_n\}$, where each $U_j \in U$ is a source-preserving union view of the form*

$$\bigcup_{i=1}^{u_j} (V_{j_i} \bowtie \{(i)\})$$

such that each V_{j_i} is a PSJR view over D and {(i)} *is a constant relation over a new attribute Source. Let $V = \{V_{1_1}, \ldots, V_{1_{u_1}}, \ldots, V_{m_1}, \ldots, V_{m_{u_m}}\}$ and let C_V be the complement of V with respect to D in accordance with Theorem* 3.22. *Let C be the set of views that is obtained from C_V by replacing each reference to a view $V_{j_i} \in V$ with the expression $\pi_{attr(V_{j_i})}(\sigma_{Source=i}(U_j))$. Then C is a super-key based complement of U with respect to D. The view inverse for $R \in D$ is defined by a variant of Equivalence* (16) *of Theorem* 3.22*, where each reference to a view $V_{j_i} \in V$ is replaced with the expression $\pi_{attr(V_{j_i})}(\sigma_{Source=i}(U_j))$.*

PROOF. Due to the form of views in U, we have $V_{j_i} \approx \pi_{attr(V_{j_i})}(\sigma_{Source=i}(U_j))$ for all $U_j \in U$. Hence, the view inverse defined by Equivalence (16) applied to complementary views in C_V gives the same result as the modified view inverse applied to complementary views in C. Consequently, C is a complement of U. By construction, C is super-key based. □

Example 3.27. Consider a banking scenario where each branch of a bank runs its own database to keep track of local customers. Let

$$D = \{db_1.\text{Customer}, \ldots, db_n.\text{Customer}\}.$$

Suppose that the view

$$U_1 =_{df} \bigcup_{i=1}^{n} (\pi_{X_1}(\sigma_\phi(db_i.\text{Customer})) \bowtie \{(i)\})$$

integrates an excerpt of this customer information. Let $U = \{U_1\}$. Assume that `CustomerID` is the key in each relation schema, that `CustomerID` $\in X_1$, and that $attr(db_1.\text{Customer}) = attr(db_i.\text{Customer})$, $1 \leq i \leq n$. To simplify notation, let $X_2 = (attr(db_1.\text{Customer}) \backslash X_1) \cup \{\texttt{CustomerID}\}$.

According to the notation of Theorem 3.26 we have $V = \{V_{1_1}, \ldots, V_{1_n}\}$, where

$$V_{1_i} =_{df} \pi_{X_1}(\sigma_\phi(db_i.\text{Customer}))$$

is a PSJR view over D, $1 \leq i \leq n$. Then, Theorem 3.22 produces the complement

$$C_V = \left\{C_{C_1}^{X_1}, C_{C_1}^{X_2}, \ldots, C_{C_n}^{X_1}, C_{C_n}^{X_2}\right\}$$

of V, where

$$C_{C_i}^{X_1} =_{df} \pi_{X_1}(db_i.\text{Customer}) \backslash \pi_{X_1}(V_{1_i}),$$
$$C_{C_i}^{X_2} =_{df} \pi_{X_2}(db_i.\text{Customer}),$$

$1 \leq i \leq n$. The view inverse is given as follows for $db_i.\text{Customer} \in D$:

$$db_i.\text{Customer} \approx \left(C_{C_i}^{X_1} \cup \pi_{X_1}(V_{1_i})\right) \bowtie C_{C_i}^{X_2}$$

In this situation, Theorem 3.26 produces the complement

$$C = \left\{C'^{X_1}_{C_1}, C_{C_1}^{X_2}, \ldots, C'^{X_1}_{C_n}, C_{C_n}^{X_2}\right\}$$

of V, where

$$C'^{X_1}_{C_i} =_{df} \pi_{X_1}(db_i.\text{Customer}) \backslash \pi_{X_1}(\sigma_{Source=i}(U_1)).$$

Moreover, the view inverse is given as follows for $db_i.\text{Customer} \in D$:

$$db_i.\text{Customer} \approx \left(C'^{X_1}_{C_i} \cup \pi_{X_1}(\sigma_{Source=i}(U_1))\right) \bowtie C_{C_i}^{X_2}$$

3.4 Complexity Results

Theorem 3.3, Corollary 3.19, and Theorems 3.22 and 3.26 provide expressions that form complements for various classes of views. In the following, we study the cost of actually *constructing* these complements, and our aim is to show that the complexity for the construction of complements is *polynomial* in the size of the underlying relation and view schemata.

Theorem 3.3 provides expressions for a minimal complement of a set V of SJR views. At the heart of this theorem lies the computation of the elementary expressions $\underline{R_j^i}$, one for each relation schema $R_j \in D$ and view $V_i \in V$. Obviously, there are $|D| \cdot |V|$ such expressions. Afterwards, these expressions are combined into one complementary view for each relation schema. Therefore, the complexity to construct the complement is polynomial in $|D| \cdot |V|$.

Corollary 3.19 provides expressions for a complement of a set V of PSJR views, based on the virtual database D^v for D. Here, a complementary view is created for each relation schema in D^v. It is easy to see that the number of attributes occurring in D, denoted by n_D, provides a rough upper bound for $|D^v|$. Thus, at most n_D complementary views are created. Moreover, each of these views can be constructed with a single scan over the views V, to check whether a view $V_i \in V$ pays a contribution or not. Therefore, the complexity to construct the complement is polynomial in $n_D \cdot |V|$.

Based on Corollary 3.19, Theorem 3.22 provides an "optimized" complement, where the number of complementary views and, hence, the number of joins

involved in the view inverses is reduced. Thus, a construction of the optimized complement can be performed as a post-processing step to the construction associated with Corollary 3.19. For this purpose, the resulting complementary views of Corollary 3.19, say C, have to be partitioned into sets that access the same warehouse views according to a similar pattern (recall Theorem 3.22). This can be achieved, for example, by comparing each view in C with every other, which requires at most $|C|^2$ steps. Keeping in mind that we have $|C| \leq n_D$ for results of Corollary 3.19, the construction of the partitioning is therefore bounded above by n_D^2. Afterwards, for each partition, a complementary view is constructed. Clearly, there are at most $|C| \leq n_D$ partitions, and the complexity to construct each view is linear in the size of the partition, which in turn is bounded above by $|C| \leq n_D$. Therefore, the complexity to construct the complement is polynomial in $n_D \cdot |V|$.

Finally, Theorem 3.26 provides expressions for a complement of a set U of source-preserving UPSJR views. For this purpose, each union view is first split into its PSJR subexpressions, which leads to a set V of PSJR views that are processed in accordance with Theorem 3.22. Afterwards, references to PSJR views in V are replaced by selections over views in U, which requires a single scan over the result of Theorem 3.22. Therefore, the complexity to construct the complement is polynomial in $n_D \cdot |V|$.

To summarize, we have:

THEOREM 3.28. *The construction of super-key based complements according to Theorem* 3.3, *Corollary* 3.19, *Theorem* 3.22, *and Theorem* 3.26 *is polynomial in the size of the underlying database schema and the view schemata as follows:*

view type	*complexity polynomial in*
SJR	$\|D\| \cdot \|V\|$
PSJR	$\|attr(D)\| \cdot \|V\|$
UPSJR	$\|attr(D)\| \cdot \|V\|$

4. DISCUSSION

In this section, we put our work in perspective by discussing a variety of related work; we start by looking into the application of data warehouses and their design, and we end with a remark on interrelational dependencies.

4.1 Complements and Self-Maintainability

View complements as discussed in this article can be exploited in the context of data warehouses to guarantee certain desirable properties, so-called *independence properties* [Laurent et al. 2001].

In short, from a technical point of view a data warehouse for one or more (operational) data sources is a separate database consisting of a set of materialized views that integrate data from the sources. For our purposes, we view a data warehouse as a set V of views over base relation schemata $D = \{R_1, \ldots R_n\}$, where different base relations may belong to different data sources. Now, it is desirable to set up data warehouses in such a way that warehouse operations, in particular its maintenance (i.e., the integration of new or updated source

data into the current warehouse instance), create little or no interferences with transactions in the underlying data sources. Clearly, for proper warehouse maintenance at least changes going on in the data sources have to be reported to the data warehouse (online or periodically). As introduced in Gupta et al. [1994], a data warehouse view V_0 in V is called *self-maintainable* if new data warehouse instances can be computed based on reported changes without additional maintainance queries, and V is self-maintainable if all its views are self-maintainable. It is easy to see that not every data warehouse is self-maintainable in general [Gupta et al. 1994], but that every warehouse can be made self-maintainable by adding a suitable amount of additional information in terms of auxiliary views [Laurent et al. 2001; Quass et al. 1996].

Moreover, given a set O of queries and/or updates over base relations, a data warehouse is *independent with respect to* O, (a) if the queries contained in O can be answered from the warehouse (without access to database relations) and (b) if—upon execution of the updates in O on the database—the warehouse can be maintained, that is, brought to a new state that reflects the updated database state, based on the update information contained in O and warehouse views (without access to database relations). Clearly, a data warehouse that is independent with respect to all database updates is self-maintainable.

The approach in Laurent et al. [2001] shows how a pre-existing warehouse V can be made independent with respect to operations O by adding a suitable portion of a view complement to V: Roughly, a complement C of V is computed first. Then, the operations O are rewritten over V and C (which is possible by using the view inverse associated with C). Finally, the complementary views appearing in the rewritings of operations O are added as auxiliary views to the original warehouse views V, which renders the data warehouse independent with respect to O.

As the computation of complements in accordance with this article generalizes the one of Laurent et al. [2001] by allowing a larger class of views, our new results can immediately be used to apply the approach of Laurent et al. [2001] for a larger class of data warehouses. Moreover, as our complements are, in general, smaller than those of Laurent et al. [2001], independence can be enforced at smaller storage costs.

We observe that the approach of Laurent et al. [2001] shows how to enforce general independence properties for *pre-existing* data warehouses and that this approach can be used for warehouse evolution when new query or update requirements come up during warehouse operation.

Alternatively, when data warehouses are designed *from scratch* the situation is slightly different. Ideally [Golfarelli and Rizzi 1998, 1999], data warehouse design proceeds—in the spirit of traditional database design—in a number of phases such as requirement analysis, conceptual design, logical design, and physical design. In particular, query requirements are automatically satisfied as a result of proper (conceptual) schema design, which implies that the warehouse is independent with respect to all "important" queries (that have been identified in the course of the requirements analysis). Thus, the remaining task with respect to independence is to render the data warehouse self-maintainable.

As explained in Lechtenbörger [2001], self-maintainability can be enforced in the course of the logical design phase (where the conceptual schema is mapped to a set of materialized views that implement the data warehouse) by (a) modifying some view definitions and (b) adding some complementary views. The key insight of Lechtenbörger [2001] lies in novel *necessary* conditions for self-maintainability of projection and join views: If a warehouse with a projection of a base relation is self-maintainable, then it is possible to determine a duplicate count for each tuple in the projection instances; if a warehouse with a join of two base relations is self-maintainable, then the information content of both relations is contained in the warehouse. Based on this insight, self-maintainability can be guaranteed by modifying projection views to include key attributes (which implies that duplicate counts are constantly 1) and adding complementary views for base relations mentioned in join views. (Notice that in case of projections [Laurent et al. 2001] adds complementary views that include the original projections and the key, which leads to redundancy among the complement and the projections. This redundancy is avoided by modifying the projection views.) As a final remark, we note that union views are always turned into source-preserving union views, which not only enforces self-maintainability for data warehouses defined by UPSJR views but also eases the task of lineage tracing [Cui and Widom 2001].

4.2 Other Related Work

The notion of relational view complement used in this paper derives directly from the more general notion of view complement first introduced by Bancilhon and Spyratos [1981], where views and complements were defined in terms of arbitrary functions over instances. Thus, Definition 2.2 is a special case of the definition of complement given in Bancilhon and Spyratos [1981]. Moreover, the view ordering $\leq_{ic}$ is exactly the view ordering given in Bancilhon and Spyratos [1981]. We note that Bancilhon and Spyratos have shown that complements always exist, but that minimal complements are almost never unique (a minimal complement of a view V is unique if and only if V is trivial, that is, if V is either constant or injective); however, the computation of complements was not addressed in Bancilhon and Spyratos [1981].

The computation of complements for views defined by relational algebra was first discussed by Cosmadakis and Papadimitriou [1984]. The approach of Cosmadakis and Papadimitriou [1984] is restricted to the setting of a single view defined by projection of a single relation, where arbitrary functional dependencies may hold. The key result of Cosmadakis and Papadimitriou [1984] states that even in this simple setting, finding a minimal complement (where a "minimal" complement is a projection with as few attributes as possible such that the view inverse is join) is NP-complete. This result does not carry over to our setting, since we do not consider arbitrary functional dependencies, but assume exactly one key to be declared for every relation schema; moreover, our complements and inverses have a different form, and our notion of minimality is completely different. On the other hand, our approach allows to compute complements for much larger classes of *sets* of views, and the above complexity

results show that the computation of complementary views is, roughly, polynomial in schema information of the data sources and the warehouse views.

Keller and Ullman [1984] have considered database instances as finite power sets and monotonic views over such database instances. In this setting, they have shown that, if for a view V there is a complement C such that V and C are independent, then C is unique. (Two views over D are independent if for all instances $V(d_1)$ and $C(d_2)$ of V and C there is an instance d_0 of D such that $V(d_0) = V(d_1)$ and $C(d_0) = C(d_2)$.) We stress that the results of Keller and Ullman [1984] do *not* carry over to our setting of super-key based complements according to Definition 2.7, because these complements are *not* monotonic in the sense of Keller and Ullman [1984] (they involve the difference operation and only the subexpression following the difference operation is monotonic). Finally, we note that the computation of complements was not addressed in Keller and Ullman [1984].

Hegner [1994] has proposed a less constraining framework than Keller and Ullman [1984] and considered database instances that are partially ordered sets with least elements and views that are monotonic morphisms (i.e., functions that preserve the least element and the partial order such that if the database instance becomes smaller so does the instance of the view). Hegner [1994] has shown that in this setting directly complemented views, that is, views that are independent components of a decomposition of the database schema, are unique if they exist and that the set of all complemented views forms a Boolean algebra. Similarly to the case of Keller and Ullman [1984], the results of Hegner [1994] do not apply to our framework, since we consider complements involving the difference operation. Finally, we note that the computation of complements was not addressed in Hegner [1994].

The expressions for the computation of super-key based complements given by Theorem 3.3, Corollary 3.19, Theorem 3.22, and Theorem 3.26 extend the results of Laurent et al. [2001] in several ways. First, complements according to Laurent et al. [2001] contain exactly one complementary view C_R per relation schema R such that $attr(R) = attr(C_R)$, whereas in the present article we allow *multiple* complementary views per relation schema R, where each complementary view is defined on a *sub*-schema of R that includes the key of R. In this way, we are able to generate smaller complements as we illustrate below. Second, we consider larger classes of views in this article; in particular, renaming and union are outside the framework of Laurent et al. [2001]. Next, we consider a different view ordering here than in Laurent et al. [2001], where query containment, $\leq_{qc}$, is used to compare sets of views. The advantage of comparing the size of views with respect to information content is that the associated ordering $\leq_{ic}$ is immediately applicable to sets of views with arbitrary schemata, whereas query containment is restricted to pairs of views that share the same schema. Finally, the complexity to construct complementary views according to the expressions given in Laurent et al. [2001] is roughly exponential in the number of warehouse views. In contrast, we have seen above that the computation of the expressions for Theorem 3.26 (which deals with the largest class of views) is polynomial in the number of database attributes and warehouse views. This surprising result is due to the fact that the computation

of complements according to Laurent et al. [2001] includes an instance of the minimal-set-cover problem (which is known to be NP-complete [Karp 1972]) to determine a set V_R of warehouse views that "covers" the schema of a relation R, whereas in our approach every warehouse view that includes a key of R is immediately used to reduce a complementary view on a sub-schema of R.

In Example 1.1, we have illustrated that the computation of complements presented in this paper can be perceived as a combination of the approaches of Cosmadakis and Papadimitriou [1984] and Laurent et al. [2001] which leads to uniformly smaller views than in both previous approaches (at lower computational complexity as we have explained above). To be precise, given a single projection view $\pi_X(R)$ defined over relation schema R (where all valid functional dependencies are implied by a key dependency), Theorem 3.22 produces exactly the minimal complement of Cosmadakis and Papadimitriou [1984], that is, a second projection such that the list of projected attributes contains $key(R)$ and all of R's attributes that are missing from X. Furthermore, Theorem 3.22 clearly leads to complements that are equal to or smaller than those computed in Laurent et al. [2001].

4.3 A Remark on Foreign Keys

As a final remark we note that the framework of Laurent et al. [2001] is based on database schemata where foreign keys can be declared. As was pointed out in Laurent et al. [2001], foreign keys imply the following kind of "optimization" of complements: If two relations are joined along a foreign key (say, a join of Employee(Id, DeptNo, ...) and Department(DeptNo, ...) along the foreign key DeptNo) then the complementary view for the relation containing the foreign key (i.e., the view for Employee) will be constantly empty (as every tuple of this relation appears in the join). We observe that this "optimization" does not require any special treatment; instead, it is automatically guaranteed by the definition of complementary views (in our framework as well as in Laurent et al. [2001]). In particular, Corollary 3.19, Theorem 3.22, and Theorem 3.26 still hold in the presence of inclusion dependencies. We point out, however, that the minimality statement of Theorem 3.3 does not hold any longer if arbitrary inclusion dependencies are allowed. To see this, consider $D = \{R_1, R_2\}$, where mutual inclusion dependencies are declared such that we have $R_1 \approx R_2$, and view $V_1 =_{df} R_1$ over D. Clearly, in this situation the empty set of views is a (minimal) complement. However, Theorem 3.3 yields a non-empty complementary view $C_{R_2} =_{df} R_2$, which is not minimal.

5. CONCLUSIONS

We have presented expressions for the computation of complements for sets of relational views. Importantly, we have shown the first minimality result concerning complements for relational views with respect to information content (Theorem 3.3). Moreover, we have given evidence that there is, in general, no hope to compute minimal complements (within the relational algebra while being useful in practice) if projection views are involved (Example 3.6); additionally, we have argued that it may not even be necessary to look for minimal

complements in practice as they are not as useful as the simpler, yet nonminimal super-key based complements (Lemmata 3.9, 3.10). Instead, we have given expressions for the computation of reasonably small complements of a large class of relational views (Theorem 3.26). Furthermore, we have shown (Theorem 3.28) that the computation of view complements is actually feasible—in spite of earlier complexity results [Cosmadakis and Papadimitriou 1984; Laurent et al. 2001], which suggested the contrary.

We conclude with a brief indication of possible applications for view complements as computed in this paper. First, view complements can be used to determine view update translators as studied in Bancilhon and Spyratos [1981]. Roughly, the problem addressed is the following. If database users issue update requests against views then these requests have to be translated into updates on database relations. It turns out, however, that such translations may not be unique and that different translations imply different update semantics. The idea of Bancilhon and Spyratos [1981] is to assign a unique semantics to view updates by choosing translations that leave a view complement unchanged. Clearly, it is easier to find a smaller complement that is left unchanged by updates than a larger one. Therefore, the refined expressions for complements given in this article enhance the chances to find translators for view updates.

Second, as we have discussed in Section 4.1, complements have found major applications in the area of data warehousing. We note that the applicability of view complements is not restricted to the standard relational data model; instead, de Amo and Halfeld Ferrari Alves [2000] show how to exploit complements to make temporal views over nontemporal sources self-maintainable.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous referees whose really careful reading and constructive remarks have lead to a significant improvement of the presentation.

REFERENCES

Abiteboul, S., Hull, R., and Vianu, V. 1995. *Foundations of Databases*. Addison-Wesley, Reading, Mass.

Aho, A. V. and Ullman, J. D. 1979. The universality of data retrieval languages. In *Proceedings of the 6th Annual Principles of Programming Languages (POPL 1979)*. 110–120.

de Amo, S. and Halfeld Ferrari Alves, M. 2000. Efficient maintenance of temporal data warehouses. In *Proceedings of the 4th IDEAS*. 188–196.

Bancilhon, F. and Spyratos, N. 1981. Update semantics of relational views. *ACM Trans. Datab. Syst. 6*, 557–575.

Chandra, A. K. and Harel, D. 1980. Computable queries for relational data bases. *J. Comput. Syst. Sci. 21*, 156–178.

Cosmadakis, S. S. and Papadimitriou, C. H. 1984. Updates of relational views. *J. ACM 31*, 742–760.

Cui, Y. and Widom, J. 2001. Lineage tracing for general data warehouse transformations. In *Proceedings of the 27th VLDB*. 471–480.

Golfarelli, M. and Rizzi, S. 1998. A methodological framework for data warehousing design. In *Proceedings of the 1st ACM DOLAP Workshop*. ACM, New York, 3–9.

Golfarelli, M. and Rizzi, S. 1999. Designing the data warehouse: Key steps and crucial issues. *J. Comput. Sci. Inf. Manage. 2*, 1–14.

GUPTA, A., JAGADISH, H. V., AND MUMICK, I. S. 1994. Data integration using self-maintainable views, Technical Memorandum, AT&T Bell Laboratories, Nov.

HEGNER, S. J. 1994. Unique complements and decomposition of database schemata. *J. Comput. Syst. Sci. 48*, 9–57.

HONEYMAN, P. 1980. Extension joins. In *Proceedings of the 6th International Conference on Very Large Data Bases (VLDB)*. 239–244.

KARP, R. M. 1972. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, R. E. Miller, Ed. Plenum Press, New York, 85–103.

KELLER, A. M. AND ULLMAN, J. D. 1984. On complementary and independent mappings on databases. In *Proceedings of the ACM SIGMOD*. ACM, New York, 143–148.

LAURENT, D., LECHTENBÖRGER, J., SPYRATOS, N., AND VOSSEN, G. 2001. Monotonic complements for independent data warehouses. *VLDB J. 10*, 295–315.

LECHTENBÖRGER, J. 2003. Data warehouse schema design. Ph.D. dissertation, University of Muenster. Available as Vol. 79 in "Dissertationen zu Datenbanken und Informationssystemen," Akademische Verlagsgesellschaft Aka GmbH, Berlin, Germany, 2001. (Extended Abstract in *Proceedings of the 10th Conference on Database Systems for Business, Technology and Web (BTW)*, 2003, 513–522).

QUASS, D., GUPTA, A., MUMICK, I. S., AND WIDOM, J. 1996. Making views self-maintainable for data warehousing. In *Proceedings of the 4th PDIS*. 158–169.

SAGIV, Y. AND YANNAKAKIS, M. 1980. Equivalences among relational expressions with the union and difference operators. *J. ACM 27*, 633–655.

SILBERSCHATZ, A., KORTH, H. F., AND SUDARSHAN, S. 2002. *Database System Concepts*, 4th ed. McGraw-Hill, New York.

VOSSEN, G. 2000. *Data Models, Database Languages and Database Management Systems*, 4th ed. (in German), Oldenbourg.

Received December 2001; revised December 2002; accepted March 2003