The VLDB Journal 10: 295–315 (2001) / Digital Object Identifier (DOI) 10.1007/s007780100055

# Monotonic complements for independent data warehouses

**D. Laurent**[1], **J. Lechtenbörger**[2,*], **N. Spyratos**[3], **G. Vossen**[2]

[1] Université de Tours, LI–IUP Informatique, 41000 Blois, France; e-mail: laurent@univ-tours.fr
[2] Universität Münster, Institut für Wirtschaftsinformatik, Leonardo-Campus 3, 48149 Münster, Germany; e-mail: {lechten,vossen}@helios.uni-muenster.de
[3] Université de Paris-Sud, LRI, CNRS-UMR 8623, Bât. 490, 91405 Orsay, France; e-mail: spyratos@lri.fr

Edited by J. Widom. Received: 21 November 2000 / Accepted: 1 May 2001
Published online: 6 September 2001–

**Abstract.** Views over databases have regained attention in the context of data warehouses, which are seen as *materialized* views. In this setting, efficient view maintenance is an important issue, for which the notion of *self-maintainability* has been identified as desirable. In this paper, we extend the concept of self-maintainability to (query and update) *independence* within a formal framework, where independence with respect to arbitrary given sets of queries and updates over the sources can be guaranteed. To this end we establish an intuitively appealing connection between warehouse independence and *view complements*. Moreover, we study special kinds of complements, namely *monotonic complements*, and show how to compute minimal ones in the presence of keys and foreign keys in the underlying databases. Taking advantage of these complements, an algorithmic approach is proposed for the specification of independent warehouses with respect to given sets of queries and updates.

**Keywords:** Data warehouse – View complement – Independence – Self-maintainability – Materialized view

## 1 Introduction

A *data warehouse* is an integrated and time-varying collection of data primarily used in organizational decision making by means of online analytical processing (OLAP) [7,28]. Typically, it is a standard database that stores *materialized views* in order to provide fast access to integrated information [28,33]. These views are extracted from multiple, heterogeneous, autonomous, distributed information sources (which are mostly operational databases), and a major difficulty lies in their proper *maintenance* [14,28]. *Incremental* view maintenance has been considered for a long time in the literature [6, 10,11,15]; an overview of maintenance of materialized views appears in [13]. In spite of those rich results, view maintenance in a warehousing environment is still complicated by the fact that the sources are decoupled from the warehouse, so that traditional incremental view maintenance may exhibit anomalies [35,36]. In this situation, the notion of *warehouse self-maintainability* has been identified as desirable. Roughly speaking, self-maintainability is the ability of a warehouse to maintain itself without "help" from the underlying databases, i.e., just based only on reported changes at the underlying databases. Self-maintainability for one view has been investigated in [2,5,12,23,25], for multiple views in [17,22], using auxiliary views in [2,22,23,25], and using conditional tables in [29]. In this paper, we generalize self-maintainability to (query and update) *independence*, and exhibit an intuitively appealing connection between warehouse independence and *view complements* [4]. In addition, we describe an algorithmic approach for the specification of independent warehouses.

We stress that, in contrast to a traditional database, a warehouse stores *integrated data*. Data integration means that data which has been extracted from the sources is merged into the warehouse, initially or after the sources have undergone updates. Integration then means: (i) materializing views of the underlying databases; and (ii) maintaining them after updates have occurred at the sources. However, maintenance is more complicated than in traditional databases for various reasons. Indeed, since the information sources are only loosely coupled to the warehouse, they do *not* participate in its maintenance; instead, they simply report their changes to the warehouse. The warehouse is typically *not* in a position to send queries back to the sources, since that can incur processing delays, the queries may be expensive, and such queries can cause warehouse maintenance anomalies [35,36]. Even worse, when information sources are highly secure or legacy systems, ad hoc queries may not be permitted at all. Consequently, it is desirable to ensure that, as much as possible, queries to the sources are not required in order to keep the warehouse data consistent [34]. Thus, the problem is how to maintain the warehouse

This work was partially supported by the bilateral French-German PROCOPE program under Grant No. 312/pro-gg; a restricted version of the problem addressed in this paper appeared in the Proc. 15th IEEE International Conference on Data Engineering 1999, 490–499.

* *Correspondence to:* J. Lechtenbörger

based on the reported changes at the sources alone. We now illustrate this problem using an example from [34].

### 1.1 Examples and motivation

*Example 1.1.* Consider the warehouse scenario shown in Fig. 1, where the warehouse consists of the single view $Sold =_{df} Sale \bowtie Emp$ over a Sales Database with relation *Sale* and a Company Database with relation *Emp* (where *clerk* is assumed to be a key for relation *Emp*).[1]

Next, let the Sales Database notify the integrator (solid arrows in Fig. 1) of the following update: "insert into *Sale* the tuple ⟨Computer, Paula⟩." As we have said, obtaining the information needed by the integrator to maintain the warehouse by querying the sources (dashed arrows) is not an option. Thus, the straightforward approach of having the Company Database join the new tuple with all tuples in relation *Emp* to find out the join tuples is not available. Instead, the warehouse should be able to *maintain itself*; to this end, notice that a subset of the *Emp* relation appears in the warehouse already (as projection of *Sold* onto *clerk* and *age*). It hence suffices to additionally keep the following information in the warehouse (where the symbol "\" denotes set difference):

$$C_{Emp} =_{df} Emp \setminus \pi_{clerk,age}(Sold)$$

Since Paula does not appear in the projection of $Sold$ onto $clerk$, a join of tuple ⟨Computer, Paula⟩ with $C_{Emp}$ now yields the data necessary to update the warehouse. Similarly, if the insertion concerns *Emp*, then the integrator will need to know the following set of tuples:

$$C_{Sale} =_{df} Sale \setminus \pi_{item,clerk}(Sold)$$

In fact, as is easily seen, $C_{Emp}$ and $C_{Sale}$ provide sufficient information for maintaining the warehouse with respect to deletions or modifications in *Sale* and *Emp* as well. □

In view maintenance, when additional queries over base data are never required to maintain a given view, the view is said to be *self-maintainable*; since a self-maintainable warehouse can be updated independently from its underlying sources, we call warehouses with that property *update-independent*.

Clearly, most warehouses are not update-independent. However, update independence can be ensured by storing additional (*auxiliary*) views at the warehouse. For instance, in our example, if we add the auxiliary views $C_{Emp}$ and $C_{Sale}$ to the warehouse, it becomes update-independent. Obviously, every warehouse can become update-independent, if *all* relevant data from the sources is copied to the warehouse. It appears to be an open problem to determine the minimum amount of extra information needed for update independence of a given warehouse [34]. We remark that the problem of self-maintainability with respect to updates, or update-independence, has attracted considerable attention in the past few years, and partial solutions for various classes of views have been proposed [22, 23, 25] (see also Sect. 4 below).

[1] In this paper, the symbol "$=_{df}$" indicates a view definition, where a relational expression is associated with a new view name; moreover, "≈" denotes view equivalence. See Sect. 2.1 below for details.

In addition, we propose to extend the concept of update independence to queries as well: indeed, there is good motivation for enabling warehouses to answer queries that could also be posed directly to the given sources. For example, sources may be unavailable or too busy to answer queries; similar to replicated databases, it may then be attractive for an application to have its queries answered from somewhere else, in this case the warehouse. Moreover, source databases might not tolerate queries from outside, or might be unable to answer queries simply because they are not databases and hence do not understand languages such as SQL or relational algebra. Intuitively, a warehouse is independent of data sources in answering source queries, or is *query-independent*, if every query to the sources can be answered using the warehouse relations *only*.

*Example 1.2.* Consider Fig. 1 once more as well as the following query to the sources:

$$q =_{df} \pi_{clerk}(Sale) \cup \pi_{clerk}(Emp)$$

(asking for all clerks that appear either in *Sale* or in *Emp*). Clearly, this query cannot be answered by the warehouse, as relation *Sold* contains only those clerks that appear in both *Sale* and *Emp*. Therefore, the warehouse of Fig. 1 is not query-independent.

However, like update independence, query independence can be ensured by storing additional (auxiliary) views at the warehouse: if we add auxiliary views $C_{Emp}$ and $C_{Sale}$ as defined in Example 1.1 to the warehouse, the warehouse becomes query-independent. Indeed, with the addition of $C_{Emp}$ and $C_{Sale}$, the warehouse becomes $\{Sold, C_{Emp}, C_{Sale}\}$ and can compute both base relations as follows:

$$Emp \approx \pi_{clerk,age}(Sold) \cup C_{Emp}$$
$$Sale \approx \pi_{item,clerk}(Sold) \cup C_{Sale}$$

In the "augmented" warehouse, query $q$ above can be answered by the following query $\overline{q}$ that uses *only* warehouse relations:

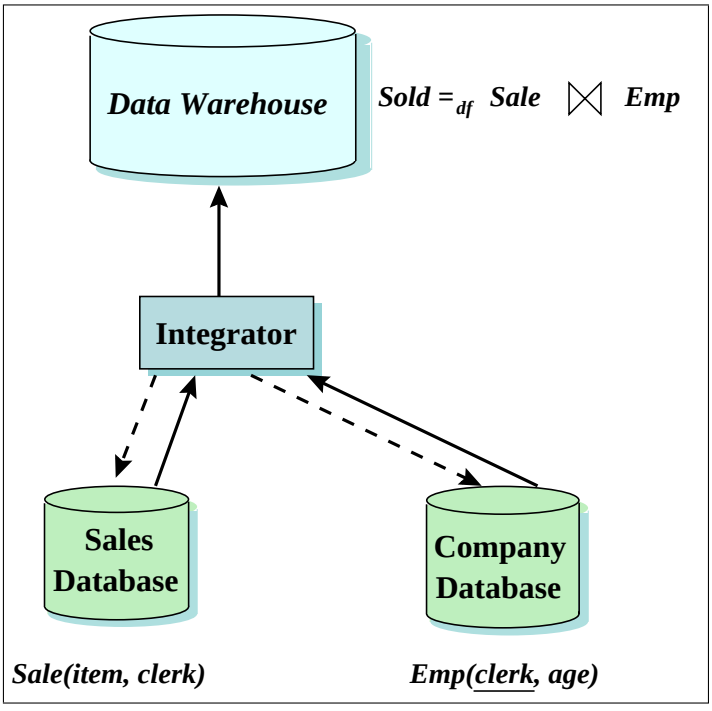
$$\overline{q} =_{df} \pi_{clerk}(Sold) \cup \pi_{clerk}(C_{Emp}) \cup \pi_{clerk}(C_{Sale}) \qquad \square$$

An important remark is in order here. In the examples that we have seen so far, we tacitly assumed that the warehouse needs to be independent with respect to *all* updates and *all* queries on base relations. In practice however, a warehouse needs to be independent only with respect to a *given set* of updates and/or queries of interest to the warehouse users. Therefore, the problem of warehouse independence that we consider in this paper can be stated as follows:

Given a data warehouse $V = \{V_1, \ldots, V_k\}$ and a set of query or update operations $OP = \{op_1, \ldots, op_m\}$ to the sources, determine a set of auxiliary views $A = \{A_1, \ldots, A_l\}$ such that the warehouse $W = V \cup A$ is independent with respect to any operation from $OP$.

We note that the above problem statement includes as a special case warehouses that need to be independent with respect to all queries and to all updates.

The reason for wanting the warehouse to be independent with respect to some but not all queries is rather obvious: the warehouse is initially designed based on a set of business queries of interest to the warehouse users. At some later time,




**Auxiliary Views:**

$$C_{Emp} =_{df} Emp \setminus \pi_{clerk,age}(Sold), \quad C_{Sale} =_{df} Sale \setminus \pi_{item,clerk}(Sold)$$

**Computation of $Emp$ and $Sale$ from views $C_{Emp}$, $C_{Sale}$, and $Sold$:**

$$Emp \approx \pi_{clerk,age}(Sold) \cup C_{Emp}, \quad Sale \approx \pi_{item,clerk}(Sold) \cup C_{Sale}$$

**Fig. 1.** Data warehouse example

however, the same warehouse users (or some new users for that matter) may become also interested in one or more queries that cannot be answered by the warehouse. Such queries will have to be answered from the sources and their processing may incur unwanted delays. It may then be interesting to have such queries answered from the warehouse and to this end we may have to store auxiliary views. Clearly, this will concern in general a few but not all queries that can possibly be answered from the sources.

Turning to updates now, it is less obvious why one would consider independence with respect to some but not all updates at the sources. Indeed, it might seem that by "ignoring" some possible updates at the sources, the warehouse may eventually become inconsistent with the sources. However, there are several situations where one may have to translate only some among the possible updates at the sources back to the warehouse, while keeping the warehouse consistent with the sources. For example, for some sources it may be the case that the warehouse uses only a small portion of just one table. In such a situation, especially when the sources are local to the warehouse, it may be preferable to simply copy periodically the updated table into the warehouse, and in this case we will have no update to translate with respect to that source. Another example is when a source is insert-only, such as the "table" storing customer transactions in a supermarket. Here again we only have to worry about the (incremental) translation of insertions and we can ignore deletions for that source. Thus, in general we need to worry about the translation of some but not all possible updates at the sources.

We conclude this motivating section by comparing our complement-based approach towards independence with two alternative naive approaches, which can be perceived as "extreme" solutions. The results of this comparison are summarized in Table 1.

The first of these naive approaches (abbreviated as RAR, for *Replicate All Relations*) is simply to replicate every relevant base relation in the data warehouse. Then the resulting warehouse is clearly independent with respect to all queries and all updates. In this case, there is exactly one materialized view per base relation, and this view is just a copy of the corresponding base relation.

**Table 1.** Approaches towards independence

| | Maintenance cost | Query cost | No. of materialized views equals/is |
|---|---|---|---|
| Replicate all base relations (RAR) | Low | High | No. of base relations |
| Materialize all queries (MAQ) | High | Low | Unbounded |
| Complement-based approach | Medium | Medium | No. of base relations + no. of warehouse views |

The RAR approach has two advantages. First, there is a *bounded* number of materialized views that have to be managed at the warehouse, regardless of user requirements. Second, maintenance of the warehouse views is relatively easy, as reported changes over base relations can be applied directly to the corresponding views, i.e., without any computations.

The main drawback of the RAR approach is a significant waste of space and time. For example, in Fig. 1 above, if the warehouse needs the set of all employees only, then storing both *Sale* and *Emp* as materialized views is a waste of warehouse space. It is also a waste of time since the view for *Sale* has to be maintained whenever a new sale is reported, *even if* such sales do not affect the set of employees.

The second naive approach (abbreviated as MAQ, for *Materialize All Queries*) is to store all user queries as materialized views.

The MAQ approach has two advantages. First, the resulting data warehouse is clearly independent with respect to all given queries. Second, query answering is as efficient as it could be, as any of the given queries can be answered by simply scanning the associated materialized view.

However, the MAQ approach has several drawbacks. First, the warehouse is not necessarily independent with respect to updates. Second, the number of materialized views that have to be maintained is potentially unbounded, as user requirements are likely to change with time. Third, the warehouse may contain redundant views. For example, in Fig. 1 above, suppose that new user needs require the materialization of the following three queries:

Q1 All employees of the *Sale* relation
Q2 All employees of the *Emp* relation
Q3 All employees that are both in the *Sale* and *Emp* relation

Clearly, the answer to Q3 can be obtained from the answers to Q1 and Q2, thus materializing Q3 leads to redundancy.

In contrast to the above naive approaches, our complement-based approach towards independence starts from a pre-existing data warehouse (which can, eventually, be empty). It then computes as few as possible auxiliary views in order to make the warehouse independent with respect to a given set of queries and updates. Therefore, our approach can be considered as a post-processing step complementing current warehouse design methods. As such, our approach supports warehouse evolution, since new (query or update) requirements can be expressed in terms of independence properties, as we have explained above.

With respect to query and maintenance costs we argue that our approach implements a reasonable trade-off between the two naive approaches. Indeed, as we will see, at most one additional view per relevant base relation is added to the data warehouse in order to ensure independence. Such a view is called a *complementary view*. Therefore, the resulting data warehouse consists of a number of pre-existing views together with a bounded number of complementary views. In this setting, we expect that most queries can be answered efficiently from pre-existing warehouse views, in fact, more efficiently than from base relations in case of RAR but less efficiently than from pre-computed views in case of MAQ. Regarding performance, since we are dealing with a bounded number of views and we exploit sharing of common parts among views, we expect maintenance costs to be considerably lower than in case of MAQ but certainly higher than in case of RAR.

### *1.2 Contributions and paper outline*

The main contributions of this paper can be summarized as follows:

1. We introduce the concept of update independence and extend it to queries, i.e., we introduce the notion of warehouse independence with respect to a given set of updates and/or queries on base relations.
2. We provide a formal framework in which warehouse independence can rigorously be defined and studied, for any number of materialized views.
3. We show how to compute a "minimal" set of auxiliary views that makes a warehouse (defined by projection, selection, and join) independent. Our computations take advantage of any key and foreign key constraints that are declared on the underlying databases.
4. We propose an algorithmic approach to the specification of independent warehouses with respect to an arbitrary set of query or update operations on base relations, and we show some uniqueness and minimality results which rely on the notion of *monotonic complement*.

We point out that the warehouse user does *not* need to be aware of auxiliary views or query translations. At warehouse definition time (or, in a running warehouse environment, even later) all necessary expressions can be automatically derived from the base relation schemes and the view definitions. Furthermore, query rewriting for answering database queries and incremental view maintenance can be integrated in the warehousing environment and can be handled automatically as well.

We note that the main focus of our contribution is the study of the *concepts* involved in warehouse independence. We are aware of the *algorithmic* problems related to our study but their treatment seems quite involved and has therefore been deferred to a forthcoming paper.

The paper is organized as follows: in Sect. 2 we give formal definitions and examples of views, complements, and monotonic complements. For views defined by projection, selection, and join we show how to compute a minimal and monotonic complement. We also show that the presence of key and foreign key constraints implies in some cases a decrease in the

size of complements. In Sect. 3 we define warehouse independence with respect to a given set of queries and/or updates, and we propose a method for obtaining independent warehouses; in particular, we prove that for warehouses defined by projection, selection, and join, for which a monotonic complement is available, under certain conditions there is exactly one subset of this complement that ensures the desired independence properties. We present related work in Sect. 4 and conclude the paper in Sect. 5.

## 2 Views and complements

We recall the basic definitions of [4] concerning views and view complements in Sects. 2.1 and 2.2, then we proceed in Sect. 2.3 to present new results for the computation of complements for views defined by projection, selection, and join.

### *2.1 Views*

We assume the reader to be familiar with the basics of relational databases, for example, along the lines of [31,32]. Throughout this paper we assume set semantics for relations and views.

We recall that a database scheme is a set of relation names, in which each relation name is associated with a set of attributes and each attribute is associated with a domain. We denote by $attr(R)$ the set of attributes associated with relation name $R$ and by $dom(A)$ the domain of attribute $A$.

For example, referring to Fig. 1, $D = \{Sale, Emp\}$ is a database scheme where

- $Sale$ and $Emp$ are relation names,
- $Sale$ is associated with the attributes $item$ and $clerk$, i.e., $attr(Sale) = \{item, clerk\}$,
- $Emp$ is associated with the attributes $clerk$ and $age$, i.e., $attr(Emp) = \{clerk, age\}$,
- $item$ and $clerk$ are associated with the domain string, and
- $age$ is associated with the domain integer.

Throughout our discussions, we consider a fixed set of relation names $D = \{R_1, \ldots, R_n\}$, possibly coming from various underlying databases. We refer to each $R_i$ as a *base relation name.* A state of $D$ has the form $d = \{(R_1, r_1), \ldots, (R_n, r_n)\}$, where $r_i$ denotes a relation over $R_i$, $1 \leq i \leq n$. Hereafter, we write $\langle r_1, \ldots, r_n \rangle$ instead of $\{(R_1, r_1), \ldots, (R_n, r_n)\}$ and refer to each $r_i$ as a *base relation.*

A *view* over $D$ is defined by a declaration of the form

$$\textit{View-name} =_{df} \textit{View-definition}$$

where *View-name* is a new relation name and *View-definition* is a relational expression over $D$. For example, referring to Fig. 1, we have the following view over $D = \{Sale, Emp\}$ :

$$Sold =_{df} Sale \bowtie Emp$$

Here $Sold$ is the view-name and $Sale \bowtie Emp$ is the view-definition[2].

[2] Actually, the view definition can be any function $f$ ranging over the states of $D$ such that $f$ computes a relation over attributes appearing in $D$ (see [4]). However, for the purposes of this paper, we restrict our attention to view definitions that are expressions from relational algebra.

We note that each relational expression is associated with a set of attributes, namely the attributes of the relation computed by the expression. Therefore, a view can be seen as a relation scheme whose name is the view name and whose attributes are those associated with the view definition. We shall refer to this scheme as the *view scheme.* For example, in Fig. 1, the expression $Sale \bowtie Emp$ is associated with the set of attributes $\{item, clerk, age\}$, therefore the view scheme is $Sold(item, clerk, age)$.

Let $U =_{df} \mathcal{E}$ be a view over $D$ where $U$ is the view name and $\mathcal{E}$ is the view definition. The state of the view scheme depends on the state of $D$ as follows: For every state $d$ of $D$ the *view state* $u$ is the relation computed by applying the expression $\mathcal{E}$ to $d$, i.e., $u = \mathcal{E}(d)$. Given a set of views $V = \{V_1, \ldots, V_k\}$, we denote by $V(d)$ the set of all view states corresponding to $d$, i.e., $V(d) = \langle V_1(d), \ldots, V_k(d) \rangle$. Moreover, following our notation, if $V$ and $C$ are sets of views such that $V = \{V_1, \ldots, V_k\}$ and $C = \{C_1, \ldots, C_l\}$, and if $d$ is a state of $D$, then

$$(V \cup C)(d) = \langle V_1(d), \ldots, V_k(d), C_1(d), \ldots, C_l(d) \rangle \ .$$

Finally, we recall the definitions of containment and equivalence of relational expressions (cf. [31,32]), and we use the notion of containment to define an ordering for views.

**Definition 2.1.** *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 *or* smaller than *$\mathcal{E}_2$, denoted by $\mathcal{E}_1 \leq \mathcal{E}_2$, if for every state $d$ of $D$ the inclusion $\mathcal{E}_1(d) \subseteq \mathcal{E}_2(d)$ holds.*
2. *$\mathcal{E}_1$ is* strictly smaller than *$\mathcal{E}_2$, denoted by $\mathcal{E}_1 < \mathcal{E}_2$, if $\mathcal{E}_1 \leq \mathcal{E}_2$ and there is a state $d$ of $D$ such that $\mathcal{E}_1(d) \subsetneq \mathcal{E}_2(d)$ holds.*
3. *$\mathcal{E}_1$ and $\mathcal{E}_2$ are* equivalent, *denoted by $\mathcal{E}_1 \approx \mathcal{E}_2$, if $\mathcal{E}_1 \leq \mathcal{E}_2$ and $\mathcal{E}_2 \leq \mathcal{E}_1$.* □

We recall that a query is defined by a relational expression; hence, Definition 2.1 immediately applies to queries. Moreover, a view is associated with a view definition, which in turn is a relational expression. Thus, given views $V_1 =_{df} \mathcal{E}_1$ and $V_2 =_{df} \mathcal{E}_2$ we say that $V_1$ is (strictly) smaller than $V_2$ (respectively, $V_1$ is equivalent to $V_2$), if $\mathcal{E}_1$ is (strictly) smaller than $\mathcal{E}_2$ (respectively, $\mathcal{E}_1$ is equivalent to $\mathcal{E}_2$).

We propose to use the following extension of the ordering "$\leq$" to sets of views: If $V = \{V_1, V_2, \ldots, V_k\}$ and $V' = \{V'_1, V'_2, \ldots, V'_k\}$ are sets of views over $D$ then $V \leq V'$ if $V_i \leq V'_i$, $1 \leq i \leq k$, holds for some ordering of the views in $V$ and $V'$.

We note that the ordering "$\leq$" is only applicable to sets of views $V$ and $V'$ with the same cardinality, i.e., with $|V| = |V'|$. However, this is not a severe restriction since the smaller of both sets, say $V'$, may be augmented with views that are constantly empty to obtain $V''$ with $|V| = |V''|$. Then the ordering of $V$ and $V'$ is defined by the ordering of $V$ and $V''$.

### *2.2 Complements*

A set $V$ of views over $D$ expresses some (but usually not all) of the information contained in $D$. Informally, any other set $C$ of views over $D$ that expresses the information "missing" from $V$ with respect to $D$ is called a *complement* of $V$. More formally, we have:

**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 $C$. □

Roughly speaking, $C$ is a complement of $V$ if each base relation of $D$ can be computed from $V$ and $C$.

Referring back to Fig. 1, we have $D = \{Sale, Emp\}$, $V = \{Sold\}$, $C_{Emp} =_{df} Emp \setminus \pi_{clerk,age}(Sold)$, and $C_{Sale} =_{df} Sale \setminus \pi_{item,clerk}(Sold)$. The set $C = \{C_{Emp}, C_{Sale}\}$ is a complement of $V = \{Sold\}$ because $Sale$ and $Emp$ can be computed from $\{Sold, C_{Emp}, C_{Sale}\}$ by the following view inverse:

$$Emp \approx \pi_{clerk,age}(Sold) \cup C_{Emp}$$
$$Sale \approx \pi_{item,clerk}(Sold) \cup C_{Sale}$$

The following proposition 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 states of $D$ to states of $V \cup C$.

**Proposition 2.1.** *Let $V$ and $C$ be two sets of views over $D$. If $C$ is a complement of $V$ then*

$$d \neq d' \text{ implies } (V \cup C)(d) \neq (V \cup C)(d')$$

*for all database states $d$ and $d'$.*

*Proof.* We show the contrapositive. Let $D = \{R_1, \ldots, R_n\}$, and let $d = \langle r_1, \ldots, r_n \rangle$ and $d' = \langle r'_1, \ldots, r'_n \rangle$ be two database states such that $(V \cup C)(d) = (V \cup C)(d')$. By Definition 2.2, there exist $n$ relational expressions $\mathcal{E}_1, \ldots, \mathcal{E}_n$ such that $R_i \approx \mathcal{E}_i$, for $i = 1, \ldots, n$. Thus, for every $i = 1, \ldots, n$, we have $r_i = \mathcal{E}_i(d)$ and $r'_i = \mathcal{E}_i(d')$. Since $\mathcal{E}_i(d)$ depends on $V \cup C$ only and since we assume that $(V \cup C)(d) = (V \cup C)(d')$, it follows that, for every $i = 1, \ldots, n$, $\mathcal{E}_i(d) = \mathcal{E}_i(d')$, which terminates the proof. □

In this paper, we restrict our attention to a specific class of views over $D$, called PSJ views, and to a specific class of complements, called monotonic complements. The notion of PSJ view is defined as follows:

**Definition 2.3.** *A* PSJ view *over $D$ is a view over $D$ whose definition has the form*

$$\pi_Z(\sigma_\phi(R_{i_1} \bowtie \ldots \bowtie R_{i_k}))$$

*where $R_{i_1}, \ldots, R_{i_k}$ are in $D$ and where $\phi$ is a selection condition.* □

Note that most SQL queries, when translated to the relational algebra, have this PSJ form.

We recall that a relational expression involving only the relational operators projection, selection, join, and union is *monotonic* [1,27]. Monotonic expressions are also relevant in this paper; indeed, they are used to define monotonic complements as follows:

**Definition 2.4.** *Let $V$ be a set of views over $D = \{R_1, \ldots, R_n\}$. Then $C$ is a* monotonic complement *of $V$, if $C$ is a complement of $V$ such that the following properties are satisfied:*

1. *$C$ is of the form $C = \{C_{R_1}, \ldots, C_{R_n}\}$, where $C_{R_i}$ is either an empty view or we have $C_{R_i} =_{df} R_i \setminus E_i$ such that $E_i$ is a monotonic relational expression over $V$, $i = 1, \ldots, n$.*
2. *The view inverse for $R_i$ is an expression over $C_{R_i}$ and $V$, $i = 1, \ldots, n$.*

*We say that $C_{R_i}$ is the* (monotonic) complementary view *for base relation $R_i$.* □

We point out that the second condition of Definition 2.4 may appear redundant at first sight. However, the following example exhibits a complement which is not monotonic, as it satisfies the first condition of Definition 2.4 but not the second.

*Example 2.1.* Let $D = \{R_1, R_2\}$, where $attr(R_1) = attr(R_2)$. Consider a warehouse $V = \{V_1, V_2\}$ 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 $C_{R_1} =_{df} R_1 \setminus V_2$ 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 $R_2 \approx (V_1 \setminus (V_2 \cup C_{R_1})) \cup V_2$. Moreover, $C_{R_1}$ and $C_{R_2}$ satisfy the first condition of Definition 2.4. However, the view inverse for $R_2$ involves $C_{R_1}$, which violates the second condition of Definition 2.4. Consequently, $C$ is not a monotonic complement. □

We note that following [27], query containment is decidable for monotonic complementary views. In particular, the tableaux techniques developed in [27] can be applied here to optimize monotonic complementary views and to remove them if they are constantly empty (i.e., if they are not necessary to compute base relations).

Referring back to Fig. 1, we have $D = \{Sale, Emp\}$ and $Sold =_{df} Sale \bowtie Emp$, which is trivially a PSJ view over $D$. Moreover, the complement is $C = \{C_{Emp}, C_{Sale}\}$, where $C_{Emp} =_{df} Emp \setminus \pi_{clerk,age}(Sold)$ and $C_{Sale} =_{df} Sale \setminus \pi_{item,clerk}(Sold)$. Finally, the inverse expressions for both relations are a union of a complementary view and a projection over $Sold$. Thus, the conditions of Definition 2.4 are met, and $C$ is a monotonic complement.

### *2.3 Computation of monotonic complements*

In the remainder of the paper, we make use of the following notation:

- Given a set $V$ of PSJ views over $D$ and $R \in D$, we denote by $V_R$ the set of views in $V$ such that $R$ appears in the view definition.
- For ease of notation, $\pi_Z(R)$ denotes the usual projection of $R$ onto attribute set $Z$ if $Z \subseteq attr(R)$, or the empty relation (over $Z$) otherwise.

In the following two sections we show how to compute monotonic complements. In Sect. 2.3.1, we consider the case where no integrity constraints are declared on the base relations $D$. In Sect. 2.3.2, we exploit the practically relevant cases of keys and foreign keys to reduce the size of the resulting monotonic complements.

#### 2.3.1 Monotonic complements in the absence of constraints

The following proposition states how to construct a monotonic complement for a set of views.

**Proposition 2.2.** *Let $D = \{R_1, \ldots, R_n\}$, and let $V = \{V_1, \ldots V_k\}$ be a set of PSJ views over $D$. Define the following views over $D$:*

$$C_{R_i} =_{df} R_i \setminus \overline{R_i}, \quad 1 \leq i \leq n \tag{1}$$

*where*

$$\overline{R_i} =_{df} \bigcup_{V_j \in V_{R_i}} \pi_{attr(R_i)}(V_j), \quad 1 \leq i \leq n\ .$$

*Then the set of views $C = \{C_{R_1}, \ldots, C_{R_n}\}$ is a monotonic complement of $V$ whose view inverse is defined by:*

$$R_i \approx C_{R_i} \cup \overline{R_i}, \quad 1 \leq i \leq n \tag{2}$$

*Proof.* As $V_j$ is a PSJ view, it follows that $\pi_{attr(R_i)}(V_j) \subseteq R_i$ for every $V_j$ in $V_{R_i}$. Therefore, $\overline{R_i} \subseteq R_i$, and since $C_{R_i} =_{df} R_i \setminus \overline{R_i}$, it follows that $R_i \approx C_{R_i} \cup \overline{R_i}$. Thus, all base relations can be computed from $V$ and $C$ as stated in Eq. (2), and hence $C$ is a complement of $V$. By construction, $C$ is monotonic. □

**Definition 2.5.** *Given a set of PSJ views $V$, the complement $C$ of $V$ defined by Proposition 2.2 is called the* canonical complement *of $V$.*

For example, referring to Fig. 1, we have:

- $D = \{Emp, Sale\}, V = \{Sold\}$
- $V_{Emp} = V_{Sale} = \{Sold\}$
- $\overline{Emp} =_{df} \pi_{clerk,age}(Sold)$ and $\overline{Sale} =_{df} \pi_{item,clerk}(Sold)$
- Therefore, the canonical complement of $\{Sold\}$ is $C = \{C_{Emp}, C_{Sale}\}$ where $C_{Emp} =_{df} Emp \setminus \pi_{clerk,age}(Sold)$ and $C_{Sale} =_{df} Sale \setminus \pi_{item,clerk}(Sold)$.
- The view inverse defined by $C$ is $Emp \approx \pi_{clerk,age}(Sold) \cup C_{Emp}$ and $Sale \approx \pi_{item,clerk}(Sold) \cup C_{Sale}$.

We note, however, that the canonical complement may not be a minimal complement, as the following example shows.

*Example 2.2.* Let $D = \{R\}$ where $attr(R) = \{A, B, C\}$ and consider the set of views $V = \{V_1, V_2, V_3\}$, where $V_1 =_{df} \pi_{AB}(R)$, $V_2 =_{df} \pi_{BC}(R)$ and $V_3 =_{df} \sigma_{B=b}(R)$. Applying Proposition 2.2, we have:

- $V_R = \{V_1, V_2, V_3\}$, since $R$ appears in every view definition.
- As $\pi_{ABC}(V_1) \approx \emptyset$, $\pi_{ABC}(V_2) \approx \emptyset$, and $V_3 \approx \pi_{ABC}(V_3)$, we obtain: $\overline{R} =_{df} V_3$.
- Therefore, the canonical complement of $V$ is $C = \{C_R\}$ where $C_R =_{df} R \setminus V_3$.
- The view inverse defined by $C$ is $R \approx C_R \cup \overline{R}$, i.e., $R \approx C_R \cup V_3$.

However, apart from the canonical complement $C = \{C_R\}$ just computed, there is a second complement $C' = \{C'_R\}$ of $V$ (which is not monotonic), where

$$C'_R =_{df} (R \bowtie \pi_{AB}((V_1 \bowtie V_2) \setminus R)) \setminus V_3\ .$$

The view inverse of $C'$ is defined by:

$$R \approx C'_R \cup V_3 \cup ((V_1 \setminus \pi_{AB}(C'_R \cup V_3)) \bowtie (V_2 \setminus \pi_{BC}(C'_R \cup V_3)))$$

Now, it is easy to see that the complement $C'$ is strictly smaller than the canonical complement $C$. Therefore the question is under what conditions the canonical complement is also a minimal complement. □

Our next goal is to prove that for SJ views (i.e., for views defined by selection and join only) the canonical complement is also a minimal complement. For this purpose we need some preliminary definitions. Given a set of views $V$ over $D$ and a database state $d$ of $D$, there are, in general, several database states $d'$ other than $d$ such that $V(d) = V(d')$. We call $d'$ *$V$-equivalent* to $d$, denoted $d' \equiv_V d$, if $V(d') = V(d)$. Given a database state $d$, the following definition designates a database state $d_r$ as the representative state in the $V$-equivalence class of $d$.

**Definition 2.6.** *Let $V$ be a set of SJ views over $D$, and let $d$ be a state of $D$. We call* representative state for $V$ and $d$ *the state $d_r$ defined by $d_r = \langle \overline{R_1}(d), \ldots, \overline{R_n}(d) \rangle$, where $\overline{R_1}, \ldots, \overline{R_n}$ are given by Eq. (1) of Proposition 2.2.* □

**Lemma 2.1.** *Let $V$ be a set of SJ views over $D$, let $d$ be a state of $D$, and let $d_r$ be the representative state for $V$ and $d$. Then we have:*

1. *$R(d_r) \subseteq R(d)$ for all $R \in D$*
2. *$V(d_r) = V(d)$*
3. *Let $d'$ be a state of $D$ such that $R(d_r) \subseteq R(d') \subseteq R(d)$ for all $R \in D$. Then we have $V(d) = V(d')$.*

*Proof.* 1 and 2 are basically well-known properties of projection-join and join-projection expressions (the selections are not essential).

3. We have to prove $V_0(d') = V_0(d)$ for all $V_0 \in V$. Let $V_0 \in V$. As $V_0$ is an SJ view, it is monotonic, hence we have $V_0(d_r) \subseteq V_0(d') \subseteq V_0(d)$. Then the equality $V(d) = V(d')$ follows from 2, which concludes the proof. □

Now we are in the position to prove that the canonical complement for SJ views is indeed minimal.

**Theorem 2.1.** *Let $V = \{V_1, \ldots, V_k\}$ be a set of SJ views over $D$. Then the canonical complement of $V$ is a minimal complement.*

*Proof.* See Appendix A.1. □

The following example illustrates the computation of canonical complements for SJ views.

*Example 2.3.* Let $D = \{R, S, T\}$ where $attr(R) = \{X, Y\}$, $attr(S) = \{Y, Z\}$ and $attr(T) = \{Z\}$. Let $V = \{V_1, V_2\}$ where $V_1 =_{df} \sigma_\phi(R \bowtie S \bowtie T)$ and $V_2 =_{df} S$. To compute the canonical complement, we apply Proposition 2.2 and we find:

- $V_R = \{V_1\}$, $V_S = \{V_1, V_2\}$ and $V_T = \{V_1\}$.
- $\overline{R} =_{df} \pi_{XY}(V_1)$, $\overline{S} =_{df} \pi_{YZ}(V_1) \cup \pi_{YZ}(V_2)$ and $\overline{T} =_{df} \pi_Z(V_1)$.
- Therefore, the canonical complement of $V$ is $C = \{C_R, C_S, C_T\}$ where

  $C_R =_{df} R \setminus \pi_{XY}(V_1)$, $C_S =_{df} S \setminus (\pi_{YZ}(V_1) \cup \pi_{YZ}(V_2))$,

  and $C_T =_{df} T \setminus \pi_Z(V_1)$.

- The view inverse defined by $C$ is:

$R \approx C_R \cup \overline{R}$, i.e., $R \approx C_R \cup \pi_{XY}(V_1)$
$S \approx C_S \cup \overline{S}$, i.e., $S \approx C_S \cup \pi_{YZ}(V_1) \cup \pi_{YZ}(V_2)$
$T \approx C_T \cup \overline{T}$, i.e., $T \approx C_T \cup \pi_{Z}(V_1)$

As $V_1$ and $V_2$ are SJ views, according to Theorem 2.1, the above complement $C$ is a minimal complement. We note that, as $\pi_{YZ}(V_2) \approx S$, we have $C_S \approx \emptyset$, i.e., the view $C_S$ is constantly empty and should not be stored as an auxiliary view. We recall that, in general, recognizing which views are constantly empty is an undecidable problem [1], whereas in our setting the techniques of [27] are applicable. □

### 2.3.2 The impact of keys and foreign keys

In this section we explore the impact of integrity constraints on the size and the form of complements. More specifically, we look into the practically relevant cases of key and foreign key constraints. We denote the key of a relation $R_i$ by $key(R_i)$ and a foreign key in relation $R_i$ with source $R_j$ by $\pi_{key(R_j)}(R_i) \subseteq \pi_{key(R_j)}(R_j)$.

Concerning key constraints we assume that at most one key is declared for every relation scheme (as is the case in SQL); if no key is declared for relation scheme $R$ then we always have $key(R) = attr(R)$.

The central ideas for the minimization of complements in the presence of constraints rely on the following observations: first and most importantly, key constraints might permit the computation of lossless joins while computing base relations from views, an observation that has been made by Honeyman [16] in an entirely different context and that has lead to the notion of an *extension join*. Second, if a foreign key $\pi_{key(R_j)}(R_i) \subseteq \pi_{key(R_j)}(R_j)$ is declared, and if the canonical complement for $V = \{R_i \bowtie R_j\}$ is computed, then the complementary view $C_{R_i}$ will always be empty, as every tuple of $R_i$ has a join partner in $R_j$; hence, the complete information concerning $R_i$ is preserved in the join.

*Example 2.4.* Refer to Fig. 1, and assume now that there is a foreign key stating that every clerk of *Sale* also appears in *Emp* (i.e., $\pi_{clerk}(Sale) \subseteq \pi_{clerk}(Emp)$). As a consequence, every tuple of *Sale* has a join partner in *Emp*. Hence $C_{Sale}$ is always empty, and we obtain $C = \{C_{Emp}, \emptyset\}$ as a complement of $V = \{Sold\}$. □

Now, we introduce some notation: let $V$ be a set of PSJ views, and let $R_j$ be a relation scheme.

- We denote by $V_{key(R_j)}$ the set of views involving $R_j$ that contain a non-trivial key of $R_j$, i.e.:

$$V_{key(R_j)} = \begin{cases} \emptyset & \text{if } key(R_j) \text{ is the trivial key} \\ \{V_i \in V_{R_j} \mid key(R_j) \subseteq attr(V_i)\} & \text{otherw.} \end{cases}$$

- We call a subset $Y$ of $V_{key(R_j)}$ such that $|Y| \geq 2$ a *cover* of $R_j$ if
  1. every attribute of $R_j$ is present in some view of $Y$, and
  2. $Y$ is minimal with respect to the above property.

  We denote by $\mathcal{C}_{R_j}^{key}$ the set of all covers of $R_j$.

Before stating our main theorem on the computation of monotonic complements, we illustrate the above notations by way of an example.

*Example 2.5.* Consider the relation schemes $R_1(A, B)$, $R_2(A, C)$, and $R_3(C, D, E)$, where $A$ is a key for $R_1$, $C$ is a key for $R_3$, $AC$ is the trivial key for $R_2$. Assume moreover that we have the following foreign keys: $\pi_A(R_2) \subseteq \pi_A(R_1)$ and $\pi_C(R_2) \subseteq \pi_C(R_3)$.

Let $V = \{V_1, V_2, V_3, V_4, V_5\}$, where $V_1 =_{df} R_1 \bowtie R_2$, $V_2 =_{df} \sigma_{\phi_2}(R_3)$, $V_3 =_{df} \pi_{CD}(\sigma_{\phi_3}(R_3))$, $V_4 =_{df} \pi_{CE}(R_3)$, $V_5 =_{df} \pi_{DE}(R_3)$. Then we have

- $V_{key(R_1)} = \{V_1\}$,
- $V_{key(R_2)} = \emptyset$,
- $V_{key(R_3)} = \{V_2, V_3, V_4\}$,
- $\mathcal{C}_{R_1}^{key} = \emptyset$,
- $\mathcal{C}_{R_2}^{key} = \emptyset$,
- $\mathcal{C}_{R_3}^{key} = \{\{V_3, V_4\}\}$.

Note that $V_{key(R_2)}$ is empty as the key of $R_2$ is trivial. Moreover, we note that $V_5$ is not contained in a cover of $R_3$, (as it does not contain the key of $R_3$). Hence, $V_5$ cannot be used to compute lossless joins. □

**Proposition 2.3.** *Let $D = \{R_1, \ldots, R_n\}$, and let $V = \{V_1, \ldots V_k\}$ be a set of PSJ views over $D$. For $1 \leq i \leq n$, define*

$$\overline{R_i} =_{df} \bigcup_{V_j \in V_{R_i}} \pi_{R_i}(V_j)$$

*and*

$$R_i^{llj} =_{df} \bigcup_{Y \in \mathcal{C}_{R_i}^{key}} \pi_{R_i}(\bowtie_{V_j \in Y} V_j).^3$$

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

$$C_{R_i} =_{df} R_i \setminus (\overline{R_i} \cup R_i^{llj}), \quad 1 \leq i \leq n, \tag{3}$$

*is a monotonic complement of $V$ whose view inverse is defined by:*

$$R_i \approx C_{R_i} \cup \overline{R_i} \cup R_i^{llj}, \quad 1 \leq i \leq n \tag{4}$$

*Proof.* The expression $\overline{R_i}$ is defined in exactly the same way as in Proposition 2.2, and so it collects all those tuples from $R_i$ which can be obtained from a view by projection. Concerning $R_i^{llj}$, we note that all joins inside the definition of $R_i^{llj}$ are along keys, therefore they are lossless and yield a subset of $R_i$. As a conclusion, every base relation can be computed from $V$ and $C$ as stated in Eq. (4), and hence $C$ is a complement of $V$. Finally, by construction, $C$ is monotonic, which completes the proof. □

**Theorem 2.2.** *Let $D$, $V$, and $C$ be as in Proposition 2.3. Then $C$ is minimal among all complements for which the computation of base relations is achieved as follows:*

1. *Joins are always performed along keys, i.e., are extension joins, and*

[3] The superscript *llj* indicates that this view contains tuples derived due to lossless joins.

2. *only complementary views and views contained in some $V_{key(R)}$ are used.*

*Proof.* Suppose that $C'$ is a complement with $C' < C$. By definition of "$<$" there are views $C_{R_i} \in C, C'_{R_i} \in C'$ and some database state $d$ such that $C'_{R_i}(d) \subset C_{R_i}(d)$. In the following let $r_i = R_i(d)$, $c_i = C_{R_i}(d)$, and $c'_i = C'_{R_i}(d)$. Since $C'$ is supposed to be a complement, in order to compute $r_i$, the tuples in $c_i \setminus c'_i$ have to be restored using views different than $C'_{R_i}$. Nevertheless, those tuples (or supertuples thereof) are not contained in any view in $V$ (by definition of $C_{R_i}$). According to the restrictions we have placed on computations here, they have to be computed using (subsets or projections of) joins using views in some $V_{key(R)}$. However, all those joins are exploited in the set $R_i^{llj}$ already. Thus, any set of views $C'$ over $D$ such that $C' < C$ is not a complement for $V$. □

Since all joins involved in the computation of a base relation are extension joins (cf. Eq. (4)), they can be performed using efficient algorithms [16].

The following example demonstrates the role of constraints in reducing the size of a complement.

*Example 2.5 (continued).* Consider again the relation schemes $R_1(A, B)$, $R_2(A, C)$, and $R_3(C, D, E)$ and views $V = \{V_1, V_2, V_3, V_4, V_5\}$. Assume first that there are no constraints. Then the covers of all base relations are empty, Eq. (3) reduces to Eq. (1), and we obtain $\overline{R_1} =_{df} \pi_{AB}(V_1)$, $\overline{R_2} =_{df} \pi_{AC}(V_1)$, $\overline{R_3} =_{df} \pi_{CDE}(V_2)$. Thus, the canonical complement $C = \{C_{R_1}, C_{R_2}, C_{R_3}\}$ is defined by $C_{R_1} =_{df} R_1 \setminus \pi_{AB}(V_1)$, $C_{R_2} =_{df} R_2 \setminus \pi_{AC}(V_1)$, and $C_{R_3} =_{df} R_3 \setminus \pi_{CDE}(V_2)$.

Assume now that $A$ is a key for $R_1$, $C$ is a key for $R_3$, $AC$ is the trivial key for $R_2$, and we have the following foreign keys: $\pi_A(R_2) \subseteq \pi_A(R_1)$ and $\pi_C(R_2) \subseteq \pi_C(R_3)$. Then, as seen previously, the covers for $R_1$ and $R_2$ are empty, whereas the covers for $R_3$ are given by $\mathcal{C}_{R_3}^{key} = \{\{V_3, V_4\}\}$. Thus, we obtain $R_1^{llj} =_{df} \emptyset$, $R_2^{llj} =_{df} \emptyset$, and $R_3^{llj} =_{df} \pi_{CDE}(V_3 \bowtie V_4)$.

Consequently, applying Proposition 2.3 the expressions for $C_{R_1}$ and $C_{R_2}$ previously computed remain unchanged and the expression for $C_{R_3}$ is now:

$$C_{R_3} =_{df} R_3 \setminus (\pi_{CDE}(V_2) \cup \pi_{CDE}(V_3 \bowtie V_4))$$

Note moreover that the join in $V_1$ is now along a foreign key which implies that $C_{R_2}$ will be constantly empty. □

### 2.3.3 Complexity results

We end the discussion on the computation of complements with some complexity results. Propositions 2.2 and 2.3 provide expressions that form a minimal complement $C$ of a set of PSJ views $V$ with respect to base relations $D$ in the absence (respectively, presence) of constraints. In the following, we study the cost of actually *constructing* $C$. Clearly, both propositions yield exactly one complementary view $C_R \in C$ per base relation $R \in D$. Thus, given $R \in D$ we next analyze the complexity to build complementary view $C_R$.

In the absence of constraints, according to Proposition 2.2, $C_R$ is given by $R \setminus \overline{R}$, where $\overline{R}$ is a union of $|V|$ views. Thus, the complexity to construct $C_R$ is linear in the size of the views in $V$.

In the presence of constraints, according to Proposition 2.3, $C_R$ is given by $R \setminus (\overline{R} \cup R^{llj})$. Clearly, the complexity of constructing $\overline{R}$ is exactly as in the case without constraints. Thus, we turn to the construction of $R^{llj}$. To construct $R^{llj}$, we have to determine the set of all covers of $R$, denoted by $\mathcal{C}_R^{key}$. For this purpose, we first look at each view $V_i \in V$ exactly once, to decide whether it contains the key of $R$. Thus, we find $V_{key(R)}$ in time $O(|V|)$. Next, the covers of $R$ have to be built starting from $V_{key(R)}$. We recall that a cover of $R$ is a subset $Y$ of $V_{key(R)}$ with $|Y| \geq 2$ such that: (a) every attribute of $R$ is present in some view of $Y$; and (b) $Y$ is minimal with respect to the above property. Now, condition (a) can be restated as follows:

$$attr(R) \subseteq \bigcup_{V_i \in V_{key(R)}} attr(V_i)$$

Clearly, this alternative formulation shows that finding a cover of $R$ is an instance of the minimal-set-cover problem, which is known to be NP-complete [18]. Consequently, the construction of $R^{llj}$, and hence the construction of $C$, is NP-complete, and we have:

**Theorem 2.3.** *The computation of minimal complements in the presence of constraints according to Proposition 2.3 is NP-complete.* □

## 3 Warehouse independence

In this section, we consider the problem of rendering a warehouse independent with respect to the underlying sources, and we provide a solution to this problem based on the notion of monotonic complement.

**Definition 3.1.** *Let $D = \{R_1, \ldots, R_n\}$. A* data warehouse *over $D$ is any set $V$ of (materialized) views over $D$.* □

We recall that a materialized view is a view whose state is physically stored in a database.

Roughly speaking, warehouse independence with respect to queries is the ability of the warehouse to answer queries posed to the underlying sources from the warehouse views. Similarly, warehouse independence with respect to updates is the ability of the warehouse to maintain itself based only on reported changes at the underlying sources (i.e., without posing any queries to the underlying sources).

A warehouse is called *independent* if it is independent with respect to queries and updates. Clearly, a warehouse is not independent in general. However, if appropriate auxiliary views are added to the warehouse, then the augmented warehouse becomes independent.

As we have explained in the Introduction, in practice a warehouse does *not* need to be independent with respect to every query and every update operation. It is sufficient that the warehouse be independent with respect to those queries and updates that are of interest to the warehouse users. Therefore, the problem of warehouse independence that we consider in this section can be stated as follows:

Given a data warehouse $V = \{V_1, \ldots, V_k\}$ over $D$ and a set of query or update operations $OP = \{op_1, \ldots, op_m\}$ over $D$, determine a set of auxiliary views $A = \{A_1, \ldots, A_l\}$ such that the warehouse $W = V \cup A$ is independent with respect to any operation from $OP$.

We note that the above problem statement includes as a special case warehouses that need to be independent with respect to *all* queries and *all* updates.

The solution that we propose in this section relies on the notion of monotonic complement. Indeed, as we shall see, the auxiliary views that provide a solution are in fact determined based on a monotonic warehouse complement.

### *3.1 Basic definitions and properties*

Let $D = \{R_1, \ldots, R_n\}$ be a set of base relation names. In the remainder of the paper,

- the term *operation* over $D$ refers to either a query or to an update over $D$ and
- we assume that keys and foreign keys may be declared over $D$.

In analogy to our notation for views, we define a query over $D$ by a declaration of the form $q =_{df} E$, where $q$ is a name and $E$ is a relational expression over $D$, and we define an update over $D$ by a declaration of the form $u =_{df} U$, where $u$ is a name and $U$ is an insertion, deletion, or modification in a base relation occurring in $D$.

We emphasize that our approach is not restricted to a specific update language; instead, in accordance with the more abstract point of view from [6], which is frequently adopted in research on view maintenance, we simply represent updates in terms of their *net effects* or *deltas*. Thus, if $u$ is an update over $D$, the effect of $u$ on an instance $d$ of $D$ is given by two "delta" relations, $\Delta^+ r$ and $\Delta^- r$, for each $R \in D$, where $\Delta^+ r$ and $\Delta^- r$ represent the tuples to be inserted into (respectively, deleted from) the instance $R(d)$ to obtain the new instance of $R$ after execution of $u$. Moreover, we assume that these delta relations are the information that is shipped from sources to the warehouse for maintenance purposes.

We next give the formal definitions of query- and update-independent warehouses.

**Definition 3.2.** *Let $V$ be a warehouse over $D$, let $q$ be a query, and let $u$ an update over $D$.*

- *$V$ is called* $q$-independent *if there is a query $\overline{q}$ over $V$ such that for all states $d$ of $D$ we have $q(d) = \overline{q}(V(d))$, i.e., if the diagram of Fig. 2 commutes.*
- *$V$ is called* $u$-independent *if there is an update $\overline{u}$ such that for all states $d$ of $D$ we have $V(u(d)) = \overline{u}(V(d))$ and $\overline{u}$ can be computed using only $u$ and $V(d)$, i.e., if the diagram of Fig. 3 commutes.*

*Let $OP$ be a set of operations over $D$. Then $V$ is called $OP$-independent, if $V$ is $op$-independent for all $op$ in $OP$.* □

The following proposition states some useful properties of query- and update-independence that follow immediately from the above definition.

**Proposition 3.1.** *Let $V_1$ and $V_2$ be two warehouses over $D$, and let $OP_1$ and $OP_2$ be two sets of operations over $D$.*

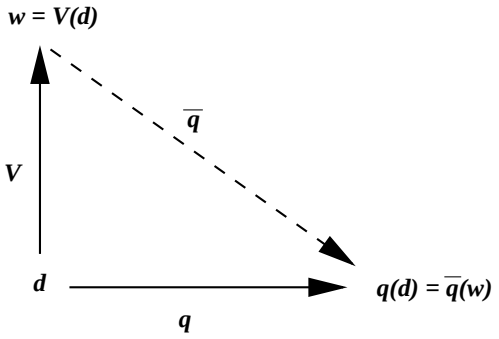



**Fig. 2.** $q$-independence expressed as a commuting diagram

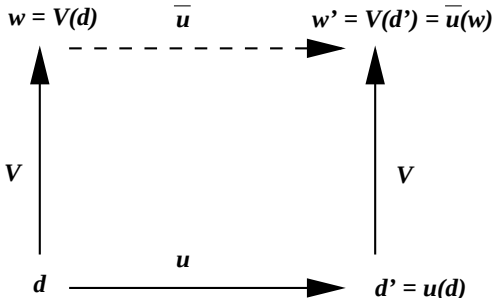



**Fig. 3.** $u$-independence expressed as a commuting diagram

1. *If $V_1$ is $OP_1$-independent and if $OP_2 \subseteq OP_1$ holds, then $V_1$ is $OP_2$-independent.*
2. *Assume that $OP_1$ contains queries only. If $V_1$ is $OP_1$-independent and if $V_1 \subseteq V_2$, then $V_2$ is $OP_1$-independent.*
3. *Assume that $OP_i$ $(i = 1, 2)$ contains queries only. If $V_i$ is $OP_i$-independent $(i = 1, 2)$, then $V_1 \cup V_2$ is $(OP_1 \cup OP_2)$-independent.*
4. *If $V_i$ $(i = 1, 2)$ is $OP_1$-independent, then $V_1 \cup V_2$ is $OP_1$-independent.*
5. *Assume that $OP_1$ contains only queries with the same set of attributes, and let $q$ be the query defined by $q =_{df} \bigcup_{q_i \in OP_1} q_i$. If $V_1$ is $OP_1$-independent, then $V_1$ is $q$-independent.* □

It is important to note that Property 2 above does not hold if $OP_1$ contains updates and Property 3 above does not hold if $OP_1$ or $OP_2$ contain updates.

For example, concerning Property 3, let $D = \{R_1, R_2\}$, and let $V = \{V_1, V_2\}$, where $V_1 = \{\sigma_\phi(R_1)\}$ and $V_2 = \{R_1 \bowtie R_2\}$. Consider $OP_1 = \{u\}$ where $u$ is an insertion into $R_1$, and $OP_2 = \{q\}$ where $q$ is the query $R_1 \bowtie R_2$. Then $V_1$ is $OP_1$-independent and $V_2$ is $OP_2$-independent. However, $V_1 \cup V_2$ is not $(OP_1 \cup OP_2)$-independent, since $q$ is not $OP_1$-independent. A similar argument can be given for Property 2.

However, as will be shown in the next section, if a warehouse $V$ contains only PSJ views together with views from a monotonic complement $C$, and if $V$ is independent with respect to a given update $u$, then adding to $V$ any view from $C$ preserves $u$-independence.

### *3.2 Warehouse independence based on monotonic complements*

We recall that, for the purposes of this paper, views are defined by projection, selection and join, and complements are monotonic complements. Thus, the problem that we consider can now be stated more precisely as follows:

> Let $V$ be a data warehouse that consists of materialized PSJ views over $D$, and let $C$ be a monotonic complement of $V$. Given a set $OP = \{op_1, \ldots, op_m\}$ of operations over $D$, determine a set $A$ of auxiliary views such that:
> 1. $A \subseteq C$,
> 2. the warehouse $W = V \cup A$ is $OP$-independent,
> 3. $A$ is minimal (with respect to set inclusion).

We provide a partial solution to this problem, proceeding as follows:

- *Auxiliary views for a single operation:* Given an operation $op$ we determine a set $A_{op}$ of auxiliary views such that $V \cup A_{op}$ is $op$-independent. Moreover, we show existence and uniqueness results concerning such a minimal set $A_{op}$ (minimality with respect to set inclusion).
- *Auxiliary views for a set of operations:* Let $A = \bigcup_{op \in OP} A_{op}$. We show that $W = V \cup A$ is $OP$-independent.
- *Complement reductions:* When $OP$ consists of PSJ queries only, then we reduce the size of complementary views in $A$ (with respect to the view ordering $\leq$).

In the following sections we present the above solution in detail.

#### 3.2.1 Auxiliary views for a single operation

Given operation $op$, we determine a set $A_{op}$ of auxiliary views such that $W = V \cup A_{op}$ is $op$-independent by applying the three steps shown in Table 2. We refer to these steps as the *independence steps.*

*Example 3.1.* Referring back to Fig. 1, we have $V = \{Sold\}$ and $C = \{C_{Emp}, C_{Sale}\}$. Suppose

$$op =_{df} \pi_{clerk}(Emp) \cap \pi_{clerk}(Sale) \ .$$

The independence steps apply as follows:

*Step 1:* The new warehouse is $W = V \cup C = \{Sold, C_{Emp}, C_{Sale}\}$.
*Step 2:* $\overline{op} =_{df} \pi_{clerk}(Sold)$
*Step 3:* As no $C_{R_i}$ appears in $\overline{op}$, we have $A_{op} = \emptyset$. □

*Example 3.2.* Next, refer again to Fig. 1 and suppose $op =_{df} \sigma_{item=VCR}(Sale)$. Now, the independence steps apply as follows:

*Step 1:* As above.
*Step 2:* $\overline{op} =_{df}$ $\pi_{item,clerk}(\sigma_{item=VCR}(Sold)) \cup \sigma_{item=VCR}(C_{Sale})$
*Step 3:* As $C_{Sale}$ appears in $\overline{op}$, whereas $C_{Emp}$ does not, we have $A_{op} = \{C_{Sale}\}$. □

*Example 3.3.* Finally, let $D = \{R\}$, let $V =_{df} \sigma_{\phi}(R)$ be a view over $D$, and consider the update $u =_{df}$ insert $\Delta r$ into $R$. Let $r$ denote the initial state of $R$.

*Step 1:* By Proposition 2.2 we have a complement $C = \{C_R\}$ of $\{V\}$, where $C_R =_{df} R \setminus \sigma_{\phi}(R)$. The view inverse defined by $C$ is given by $R \approx V \cup C_R$. The new warehouse is $W = \{V, C_R\}$.
*Step 2:* We translate the update using the view inverse as follows:

$$\begin{aligned} \overline{u}(w) &= W(u(W^{-1}(V(r), C_R(r)))) \\ &= W(u(V(r) \cup C_R(r))) \\ &= W(V(r) \cup C_R(r) \cup \Delta r) \end{aligned}$$

As $W$ consists of $V$ and $C_R$, we have:

$$\begin{aligned} & W(V(r) \cup C_R(r) \cup \Delta r) \\ &= \langle V((V(r) \cup C_R(r) \cup \Delta r)), \\ &\quad C_R((V(r) \cup C_R(r) \cup \Delta r))\rangle \\ &= \langle \sigma_{\phi}((\sigma_{\phi}(r) \cup C_R(r) \cup \Delta r)), \\ &\quad C_R((\sigma_{\phi}(r) \cup C_R(r) \cup \Delta r))\rangle \\ &= \langle \sigma_{\phi}(r) \cup \sigma_{\phi}(\Delta r), C_R((\sigma_{\phi}(r) \cup C_R(r) \cup \Delta r))\rangle \\ &= \langle \sigma_{\phi}(r) \cup \sigma_{\phi}(\Delta r), C_R(r) \cup \sigma_{\neg\phi}(\Delta r)\rangle \end{aligned}$$

We remark that one has to observe that $C_R(r) = \sigma_{\neg\phi}(r)$ and $V(C_R(r)) = C_R(V(r)) = \emptyset$ to obtain the above equations. To summarize, we have: $\overline{u}(w) = \langle V(r) \cup \sigma_{\phi}(\Delta r), C_R(r) \cup \sigma_{\neg\phi}(\Delta r)\rangle$
*Step 3:* As no $C_{R_i}$ appears in the computation of the new state for $V$ (which is given by $V(r) \cup \sigma_{\phi}(\Delta r)$), we have $A_u = \emptyset$. □

**Theorem 3.1.** *Let $V$ be a warehouse over $D$, and let $C$ be a monotonic complement of $V$. Let $op$ be an operation over $D$, and let $A_{op}$ be the set of auxiliary views produced according to the independence steps. Then $W = V \cup A_{op}$ is an $op$-independent warehouse.*

*Proof.* See Appendix A.2. □

It is important to note from Example 3.3 that the set $A_{op}$ is not unique, mainly depending on how the expression in Step 2 is produced. Indeed, if one does not notice that $V(C_R(r)) = \emptyset$, then the set $A'_u$ produced at Step 3 is $\{C_R\}$. In this case, although $V \cup \{C_R\}$ is $u$-independent, it is not necessary to store $C_R$.

The following theorem, however, states that there is always a minimal subset $A_{op}$ of $C$ that ensures $op$-independence. In addition, this set $A_{op}$ is even unique if we apply the independence steps to a *monotonic* complement and there are no foreign keys declared over $D$.

**Theorem 3.2.** *Let $V$ be a warehouse over $D$, let $C$ be a monotonic complement of $V$, and let $op$ be an operation over $D$.*

1. *Then there is a subset $A_{op}$ of $C$ such that*
   *(a) $V \cup A_{op}$ is $op$-independent and*
   *(b) $A_{op}$ is minimal with respect to set inclusion, i.e., if we remove any of the views in $A_{op}$ then (a) above does not hold.*

**Table 2.** The independence steps

| |
|---|
| *Step 1:* Find a monotonic complement $C$ of $V$ and add it to $V$, thus obtaining a new warehouse $W = V \cup C$.<br>Note that we do not materialize $C$ at this step; we just use it for subsequent computations.<br>In addition, note that according to Proposition 2.1, there is now a one-to-one mapping from database states to warehouse states.<br>We denote this mapping by $W$ and its inverse by $W^{-1}$. |
| *Step 2:* Translate $op$ into a query or update $\overline{op}$ over the warehouse as follows:<br>• If $op$ is a query, then define the query $\overline{op}$ over $W$ by:<br>$\overline{op}(w) = op(W^{-1}(w))$, for all states $w$ of $W$.<br>• If $op$ is an update, then define the update $\overline{op}$ over $W$ by:<br>$\overline{op}(w) = W(op(W^{-1}(w)))$, for all states $w$ of $W$. |
| *Step 3:* Define $A_{op}$ as follows:<br>• If $op$ is a query, then define $A_{op}$ to be the set of $C_{R_i}$'s appearing in the expression of $\overline{op}$.<br>• If $op$ is an update, then define $A_{op}$ to be the of $C_{R_i}$'s appearing in the maintenance expressions for views in $V$ as given by $\overline{op}$. |

2. *If there are no foreign keys declared over $D$ then the minimal set $A_{op}$ mentioned in (1) is unique.*

*Proof.* See Appendix A.3. □

We point out that this theorem holds for warehouses defined by PSJ views, but not for arbitrary views. First, the theorem makes use of monotonic complements, and it is not clear how monotonic complements can be computed for a larger class of views. Second, the following examples shows that there is no hope to preserve uniqueness when starting from arbitrary views where a (monotonic) complement is derived in an ad hoc manner or when starting from base relations where arbitrary foreign keys may hold.

*Example 3.4.* As in Example 2.1 let $D = \{R_1, R_2\}$, where $attr(R_1) = attr(R_2)$, and consider a warehouse $V = \{V_1, V_2\}$ over $D$, where $V_1 =_{df} R_1 \cup R_2$ and $V_2 =_{df} R_1 \cap R_2$. Let $op =_{df} R_1$. Clearly, some auxiliary information is required for $op$-independence. Let $C = \{C_{R_1}, C_{R_2}\}$, where $C_{R_1} =_{df} R_1 \setminus V_2$ and $C_{R_2} =_{df} R_2 \setminus V_2$. It is easy to see that $C$ is a complement of $V$.

However, $\{C_{R_1}\}$ and $\{C_{R_2}\}$ are also complements of $V$. As we have already seen in Example 2.1, we have $R_1 \approx V_2 \cup C_{R_1}$ and $R_2 \approx (V_1 \setminus (V_2 \cup C_{R_1})) \cup V_2$, which shows that $\{C_{R_1}\}$ is a complement of $V$; the proof for $\{C_{R_2}\}$ is similar. Thus, to guarantee $op$-independence, either $R_1 \setminus V_2$ or $R_2 \setminus V_2$ may be added to $V$.

Next, consider $D$ from above and assume that we have $key(R_1) = attr(R_1) = attr(R_2) = key(R_2)$, as well as the cyclic foreign keys $\pi_{key(R_1)}(R_1) \subseteq \pi_{key(R_1)}(R_2)$ and $\pi_{key(R_1)}(R_2) \subseteq \pi_{key(R_1)}(R_1)$, which imply $R_1 \approx R_2$. Clearly, any query involving either $R_1$ or $R_2$ can be answered from each individual base relation. Consequently, if $V'$ is a set of PSJ views over $D$, $C'$ is a monotonic complement of $V'$, and $op$ is an operation over $D$ such that $V'$ is not $op$-independent, then we can either add the complementary view for $R_1$ or the complementary view for $R_2$ to enforce $op$-independence. □

### 3.2.2 Auxiliary views for a set of operations

Now we turn to the general case when $OP$ consists not just of one operation but of any set of operations. More precisely, let $OP$ be a set of operations over $D$, let $V$ be a warehouse over $D$, and let $C$ be a monotonic complement of $V$. Let $A_{OP} = \bigcup_{op \in OP} A_{op}$, where $A_{op}$ is the set of auxiliary views produced by the independence steps for operation $op \in OP$. We show that $V \cup A_{OP}$ is $OP$-independent.

**Lemma 3.1.** *Let $V$ be a warehouse over $D$, and let $C$ be a monotonic complement of $V$. Let $op$ be an update over $D$, and let $A_{op}$ be the subset of $C$ as determined by the independence steps for $op$-independence.*

*Then for every subset $C'$ of $C$, the warehouse $V \cup A_{op} \cup C'$ is $op$-independent.*

*Proof.* The proof proceeds exactly as the proof of Theorem 3.1. □

**Theorem 3.3.** *Let $V$ be a warehouse over $D$, and let $C$ be a monotonic complement of $V$. Let $OP$ be a set of operations over $D$, and let $A_{OP} = \bigcup_{op \in OP} A_{op}$, where $A_{op}$ is the set of auxiliary views produced by the independence steps for operation $op \in OP$. Then $W = V \cup A_{OP}$ is an $OP$-independent warehouse.*

*Proof.* We show the theorem in the case $m = 2$, from which the general case follows. If $op_1$ and $op_2$ are both queries, then the result follows immediately from Proposition 3.1 (3).

Assume now that $op_1$ is an update. Then $V \cup A_{op_1}$ is $op_1$-independent, and thus, by Lemma 3.1 (since $A_{op_2}$ contains views from a monotonic complement), $W = V \cup A_{op_1} \cup A_{op_2}$ is $op_1$-independent. Assume moreover that $op_2$ is a query, then Proposition 3.1 (2) shows that $W$ is also $op_2$-independent. Assume now that $op_2$ is an update. Since $V \cup A_{op_2}$ is $op_2$-independent, then, by Lemma 3.1, so is $W$. Thus we obtain that $W$ is $OP$-independent, which terminates the proof. □

So far, we have shown that the independence steps (which are formulated for individual queries or updates) can be exploited to guarantee $OP$-independence for a set of operations $OP$ in a natural way.

As mentioned previously, the independence steps do not always lead to minimal auxiliary sets. However, as a consequence of Theorem 3.2 and Theorem 3.3 we can show that if each $A_{op}$ is a unique minimal set, then so is the set $A_{OP}$.

**Theorem 3.4.** *Let $V$ be a warehouse over $D$, and let $C$ be a monotonic complement of $V$. Let $OP$ be a set of operations over $D$, and let $A_{OP} = \bigcup_{op \in OP} A_{op}$, where $A_{op}$ is the set of auxiliary views produced by the independence steps for operation $op \in OP$.*

*If, for every $op \in OP$, $A_{op}$ is the unique minimal subset of $C$ such that $V \cup A_{op}$ is $op$-independent, then $A_{OP}$ is the*

*unique minimal subset of $C$ such that $W = V \cup A_{OP}$ is $OP$-independent.*

*Proof.* $OP$-independence follows from Theorem 3.3.

Concerning minimality and uniqueness, let $A'_{OP}$ be a minimal subset of $C$ such that $W = V \cup A'_{OP}$ is $OP$-independent. Assume that there is an auxiliary view $C_R$ such that $C_R \in A_{OP} \setminus A'_{OP}$. Then there exists $op \in OP$ such that $C_R \in A_{op}$; moreover, $V \cup A'_{OP}$ is $op$-independent. Let $A'_{op}$ be a minimal subset of $A'_{OP}$ that ensures $op$-independence. Thus we have $C_R \in A_{op} \setminus A'_{op}$, showing that $A_{op}$ and $A'_{op}$ are different sets. This is in contradiction to the uniqueness assumption. Consequently, we have $A_{OP} = A'_{OP}$, which concludes the proof. □

### 3.2.3 Complement reductions

So far, we have concentrated on the question of guaranteeing $OP$-independence by using a minimal subset $A_{OP}$ of a monotonic complement. However, $A_{OP}$ might contain information that is not essential for $OP$-independence. We illustrate this observation for the query $\sigma_{item=VCR}(Sale)$ whose independence steps have been computed in Example 3.2.

*Example 3.5.* Recall from Fig. 1 that we have base relations *Sale(item, clerk)* and *Emp(clerk, age)*, and the warehouse $V = \{Sold\}$ where $Sold =_{df} Sale \bowtie Emp$. If $op$ is the query $\sigma_{item=VCR}(Sale)$, then the associated independence steps give $A_{op} = \{C_{Sale}\}$ where $C_{Sale} =_{df} Sale \setminus \pi_{item,clerk}(Sold)$.

In general, however, $C_{Sale}$ contains tuples where *item = VCR* does not hold, although these tuples are not necessary for $op$-independence. In fact, a smaller amount of auxiliary information, which is sufficient and necessary for $op$-independence, is $C'_{Sale} =_{df} \sigma_{item=VCR}(C_{Sale})$. □

To account for this observation, we now improve Theorem 3.3 using traditional push-down rules for query optimization. Any PSJ query $op =_{df} \pi_Z(\sigma_\phi(R_1 \bowtie \ldots \bowtie R_k))$ is equivalent to a union of PSJ expressions of the form

$$q =_{df} \pi_Z(\sigma_{\phi_0}(\sigma_{\phi_1}(R_1) \bowtie \ldots \bowtie \sigma_{\phi_k}(R_k))),$$

where

- $\phi_0$ is a conjunction of non-local atomic selection conditions and
- $\phi_i$ is a conjunction of local atomic selection conditions, $1 \leq i \leq k$.

Let $q^{conj}$ be the set of PSJ expressions $q$ for $op$. Then by Proposition 3.1 (5), a warehouse is $op$-independent if it is $q^{conj}$-independent.

Moreover, we recall that if $op$ is a PSJ query $op =_{df} \pi_Z(\sigma_\phi(R_1 \bowtie \ldots \bowtie R_k))$, then the set $A_{op}$ produced by the independence steps is a subset of $\{C_{R_1}, \ldots, C_{R_k}\}$, say $A_{op} = \{C_{R_{i_1}}, \ldots, C_{R_{i_l}}\}$ for $l \leq k$. To simplify the following presentation, we assume that the independence steps always produce an auxiliary view for each base relation, where an auxiliary view may be empty.

In the following proposition, we provide an optimized set of auxiliary views for a PSJ query where the selection condition is a conjunction. Then, based on the above considerations, we extend the result to a set of PSJ queries having any selection condition. To state our result, we need the operation of a *semijoin*, defined as follows: $r_i \ltimes r_j = \pi_{attr(R_i)}(r_i \bowtie r_j)$.

**Proposition 3.2.** *Let $V$ be a warehouse over $D$, and let $C$ be a monotonic complement of $V$. Let $op =_{df} \pi_Z(\sigma_\phi(R_1 \bowtie \ldots \bowtie R_k))$ be a PSJ query over $D$ where $\phi = \phi_0 \wedge \phi_1 \wedge \ldots \wedge \phi_k$ is a conjunctive selection condition such that $\phi_i$ is the conjunction of all atomic conditions in $\phi$ related to $R_i$ ($i = 1, \ldots, k$) and $\phi_0$ is the conjunction of all non-local atomic conditions in $\phi$.*

*Let $A_{op} = \{C_{R_1}, \ldots, C_{R_k}\}$ be the set of auxiliary views produced by the independence steps, and let $C'_{R_i}$ ($i = 1, 2, \ldots, k$) be defined by:*

$$C'_{R_i} =_{df} \sigma_{\phi_i}(C_{R_i}) \ltimes \left(\sigma_{\phi_0}\left(\bowtie_{i=1}^{k} \sigma_{\phi_j}(R_j)\right)\right)$$

*Let $A'_{op} = \{C'_{R_1}, \ldots, C'_{R_k}\}$. Then $W = V \cup A'_{op}$ is $op$-independent.*

*Proof.* The proof comes from the fact that we keep in each auxiliary view $C'_i$ only those tuples from $C_i$ that $(i)$ satisfy the selection condition $\phi_i$ and that $(ii)$ participate in the join in $op$. □

*Example 3.5 (continued).* By Proposition 3.2, we obtain $C'_{Sale} =_{df} \sigma_{item=VCR}(C_{Sale})$. □

Now, let $op =_{df} \pi_Z(\sigma_\phi(R_1 \bowtie \ldots \bowtie R_k))$ be a PSJ query where $\phi$ is any selection condition. Then, $op$ can be written as $op^1 \cup \ldots \cup op^l$ where every $op^j$ is a PSJ query with conjunctive selection condition. Applying Proposition 3.2 above to every $op^j$, we obtain sets $A^j = \{C^j_1, \ldots, C^j_k\}$ such that $V \cup A^j$ is $op^j$-independent. Let

$$A'_{op} = \left\{ \bigcup_{j=1}^{l} C^j_1, \ldots, \bigcup_{j=1}^{l} C^j_k \right\}.$$

By Proposition 3.1 (1,3), $V \cup A'_{op}$ is $op^j$-independent, for every $j = 1, \ldots, l$, which implies, by Proposition 3.1 (5), that $V \cup A'_{op}$ is $op$-independent.

Let us now consider $OP = \{op_1, \ldots, op_m\}$, a set of PSJ queries over $D$, and let us denote by $A'_i$ the set of auxiliary views associated to $op_i$ as obtained just above. By Proposition 3.1 (3), $V \cup A'_{OP}$ where $A'_{OP} = A'_1 \cup \ldots \cup A'_m$ is $OP$-independent. If we denote by $C''_{R_i}$ the union of all auxiliary views in $A'_{OP}$ which are subsets of the complementary view $C_{R_i}$ in a monotonic complement $C$ of $V$, if we use $A''_{OP}$ to denote the set of these union views $C''_{R_i}$, then Proposition 3.1 implies that $V \cup A''_{OP}$ is $OP$-independent. Therefore, we have the following theorem which in conjunction with Proposition 3.2 shows that we are able to minimize the complement *in theory*, according to our ordering of the views. Nevertheless, the *computation* (and thus the maintenance of that minimal complement) may still be expensive.

**Theorem 3.5.** *Let $V$ be a warehouse over $D$, and let $C$ be a monotonic complement of $V$. Let $OP$ be a set of PSJ queries over $D$, and let $A''_{OP}$ be the set of auxiliary views as defined above. Then $V \cup A''_{OP}$ is $OP$-independent.* □

*Example 3.6.* Consider

$$D = \{R(A, B), S(A, C), T(A, D)\},$$

and a warehouse $V = \{V_1, V_2\}$, where

$V_1 =_{df} \pi_{A,B}(\sigma_{C<0}(R \bowtie S))$,
$V_2 =_{df} \sigma_{D\geq 0}(T)$.

Let $OP = \{op_1, op_2, op_3\}$, where

$op_1 =_{df} \sigma_{(B=D)\vee(A\leq 7)}(R \bowtie T)$,
$op_2 =_{df} \pi_A(\sigma_{D>7}(T))$,
$op_3 =_{df} \pi_{A,C}(\sigma_{B=5}(R \bowtie S))$.

Then the canonical complement of $V$ is $C = \{C_R, C_S, C_T\}$ where $C_R =_{df} R \setminus V_1$, $C_S =_{df} S$, and $C_T =_{df} T \setminus V_2$ and it is easy to see that $A_1 = \{C_R, C_T\}$, $A_2 = \emptyset$ (because $\sigma_{D>7}(T)$ always gives a subset of $V_2$) and $A_3 = \{C_S\}$. Therefore, $A_{OP} = \{C_R, C_S, C_T\}$. On the other hand, using Theorem 3.5, we can reduce the size of all auxiliary views in $A_{OP}$ as follows:

- First $op_1$ is decomposed into $op_{1_1} =_{df} \sigma_{B=D}(R \bowtie T)$ and $op_{1_2} =_{df} \sigma_{A\leq 7}(R) \bowtie \sigma_{A\leq 7}(T)$ and $op_3$ is written as $\sigma_{B=5}(R) \bowtie S$.
- Applying Proposition 3.2, we obtain:
  – $A'_{1_1} = \{C_R \ltimes (\sigma_{B=D}(R \bowtie T)),$
  $C_T \ltimes (\sigma_{B=D}(R \bowtie T))\}$
  – $A'_{1_2} = \{\sigma_{A\leq 7}(C_R) \ltimes (\sigma_{A\leq 7}(R) \bowtie \sigma_{A\leq 7}(T)),$
  $\sigma_{A\leq 7}(C_T) \ltimes (\sigma_{A\leq 7}(R) \bowtie \sigma_{A\leq 7}(T))\}$
  – $A'_2 = \emptyset$, since $A_2$ is empty as well
  – $A'_3 = \{C_S \ltimes (\sigma_{B=5}(R) \bowtie S)\}$.
- Therefore, $A''_{OP} = \{C''_1, C''_2, C''_3\}$ is defined by:

$$C''_1 =_{df} (C_R \ltimes (\sigma_{B=D}(R \bowtie T))) \cup (\sigma_{A\leq 7}(C_R) \ltimes (\sigma_{A\leq 7}(R) \bowtie \sigma_{A\leq 7}(T)))$$
$$C''_2 =_{df} C_S \ltimes (\sigma_{B=5}(R) \bowtie S)$$
$$C''_3 =_{df} (C_T \ltimes (\sigma_{B=D}(R \bowtie T))) \cup (\sigma_{A\leq 7}(C_T) \ltimes (\sigma_{A\leq 7}(R) \bowtie \sigma_{A\leq 7}(T)))$$

□

### *3.3 A detailed example*

In this section, we work out a detailed example, which demonstrates the following strengths of our approach:

- Computation of complements for *sets* of views.
- Computation of *minimal* complements by taking keys and foreign keys into account.
- $OP$-Independence with respect to *arbitrary* sets of operations $OP$.
- Derivation of a *minimal* set of auxiliary views to ensure $OP$-independence.

The example that we use builds on our running example of Fig. 1 and considers a database scheme $D = \{Emp, Catalog, Sale\}$, where

- $attr(Emp) = \{clerk, age\}$,
- $attr(Catalog) = \{item, price, description\}$,
- $attr(Sale) = \{item, clerk, date, amount\}$,
- $key(Emp) = \{clerk\}$,
- $key(Catalog) = \{item\}$,
- $key(Sale) = \{item, clerk, date\}$,
- $\pi_{item}(Sale) \subseteq \pi_{item}(Catalog)$, and
- $\pi_{clerk}(Sale) \subseteq \pi_{clerk}(Emp)$.

We assume a warehouse $V = \{ItemDesc, GoodSales, YoungSales\}$ over $D$ whose views are defined as follows:

$$ItemDesc =_{df} \pi_{item,description}(Catalog)$$
$$GoodSales =_{df} \pi_{item,clerk,date,amount,price}(\sigma_{price>1000}(Sale \bowtie Catalog))$$
$$YoungSales =_{df} \sigma_{age<30}(Sale \bowtie Emp)$$

We want this warehouse to be independent with respect to three queries ($op_1$, $op_2$, and $op_3$) and two updates ($op_4$ and $op_5$, which are specified in terms of delta relations) defined as follows:

$op_1 =_{df} \pi_{clerk,price}(\sigma_{price>10000}(Sale \bowtie Catalog))$
$op_2 =_{df} \pi_{item,price}(Catalog)$
$op_3 =_{df}$
$\pi_{clerk}(Emp) \setminus (\pi_{clerk}(\sigma_{price>10000}(Sale \bowtie Catalog)))$
$op_4 =_{df}$ insert $\Delta e$ into $Emp$
$op_5 =_{df}$ insert $\Delta s$ into $Sale$

In other words, we want the warehouse $V$ to be $OP$-independent where $OP = \{op_1, op_2, op_3, op_4, op_5\}$. To this end we apply the independence steps as follows:

*Step 1.* We apply Proposition 2.3 to obtain a monotonic complement $C$ of $V$ with respect to $D$. Note that, according to Theorem 2.2, $C$ is minimal.

$$\overline{Emp} =_{df} \pi_{attr(Emp)}(YoungSales)$$
$$Emp^{llj} =_{df} \emptyset$$
$$\overline{Catalog} =_{df} \emptyset$$
$$Catalog^{llj} =_{df} \pi_{attr(Catalog)}(GoodSales \bowtie ItemDesc)$$
$$\overline{Sale} =_{df} \pi_{attr(Sale)}(YoungSales) \cup \pi_{attr(Sale)}(GoodSales)$$
$$Sale^{llj} =_{df} \emptyset$$

Therefore, we obtain:

$$C_{Emp} =_{df} Emp \setminus \overline{Emp}$$
$$C_{Catalog} =_{df} Catalog \setminus Catalog^{llj}$$
$$C_{Sale} =_{df} Sale \setminus \overline{Sale}$$

Thus, $C = \{C_{Emp}, C_{Catalog}, C_{Sale}\}$ is a monotonic minimal complement of $V$ and has the following view inverse:

$$Emp \approx C_{Emp} \cup \overline{Emp}$$
$$Catalog \approx C_{Catalog} \cup Catalog^{llj}$$
$$Sale \approx C_{Sale} \cup \overline{Sale}$$

We remark that the size of $C_{Catalog}$ is reduced by exploiting a lossless join in $Catalog^{llj}$, which is possible due to key constraints. Furthermore, the computation of $C_{Sale}$ takes advantage of multiple views involving $Sale$.

*Step 2.* One by one, we translate the operations occurring in $OP$ using the view inverses associated with $C$.

Concerning $op_1$ we have:

$$\begin{aligned}
op_1 =_{df}\ & \pi_{clerk,price}(\sigma_{price>10000}(Sale \bowtie Catalog)) \\
\approx\ & \pi_{clerk,price}(\sigma_{price>10000}((C_{Sale} \cup \overline{Sale}) \\
& \quad \bowtie (C_{Catalog} \cup Catalog^{llj}))) \\
\approx\ & \pi_{clerk,price}((C_{Sale} \cup \overline{Sale}) \\
& \quad \bowtie \sigma_{price>10000}(C_{Catalog} \cup Catalog^{llj})) \\
\approx\ & \pi_{clerk,price}((C_{Sale} \cup \overline{Sale}) \\
& \quad \bowtie \sigma_{price>10000}(Catalog^{llj})) \\
\approx\ & \pi_{clerk,price}(\overline{Sale} \bowtie \sigma_{price>10000}(Catalog^{llj})) \\
\approx\ & \pi_{clerk,price}((\pi_{attr(Sale)}(YoungSales) \\
& \quad \cup \pi_{attr(Sale)}(GoodSales)) \\
& \quad \bowtie\ \sigma_{price>10000}(Catalog^{llj})) \\
\approx\ & \pi_{clerk,price}(\pi_{attr(Sale)}(GoodSales) \\
& \quad \bowtie \sigma_{price>10000}(Catalog^{llj})) \\
\approx\ & \pi_{clerk,price}(\sigma_{price>10000}(GoodSales))
\end{aligned}$$

Therefore, we obtain

$$\overline{op_1} =_{df} \pi_{clerk,price}(\sigma_{price>10000}(GoodSales)),$$

which shows that no complementary view is necessary to rewrite $op_1$ over $V \cup C$. As a consequence, we have $A_{op_1} = \emptyset$.

Concerning $op_2$ we have:

$$\begin{aligned}
op_2 =_{df}\ & \pi_{item,price}(Catalog) \\
\approx\ & \pi_{item,price}(C_{Catalog} \cup Catalog^{llj}) \\
\approx\ & \pi_{item,price}(C_{Catalog}) \cup \pi_{item,price}(Catalog^{llj}) \\
\approx\ & \pi_{item,price}(C_{Catalog}) \cup \pi_{item,price}( \\
& \quad \pi_{attr(Catalog)}(GoodSales \bowtie ItemDesc))
\end{aligned}$$

Therefore, we obtain

$$\begin{aligned}
\overline{op_2} =_{df}\ & \pi_{item,price}(C_{Catalog}) \cup \pi_{item,price}( \\
& \quad \pi_{attr(Catalog)}(GoodSales \bowtie ItemDesc)),
\end{aligned}$$

which shows that $C_{Catalog}$ is necessary to rewrite $op_2$ over $V \cup C$. As a consequence, we have $A_{op_2} = \{C_{Catalog}\}$.

Concerning $op_3$ we have:

$$\begin{aligned}
op_3 =_{df}\ & \pi_{clerk}(Emp) \setminus \\
& \quad (\pi_{clerk}(\sigma_{price>10000}(Sale \bowtie Catalog))) \\
\approx\ & \pi_{clerk}(C_{Emp} \cup \overline{Emp}) \setminus \\
& \quad \pi_{clerk}(\sigma_{price>10000}((C_{Sale} \cup \overline{Sale}) \\
& \quad\ \bowtie (C_{Catalog} \cup Catalog^{llj}))) \\
\approx\ & \pi_{clerk}(C_{Emp} \cup \overline{Emp}) \setminus \\
& \quad \pi_{clerk}(\sigma_{price>10000}(GoodSales)) \\
\approx\ & (\pi_{clerk}(C_{Emp}) \setminus \\
& \quad\ \pi_{clerk}(\sigma_{price>10000}(GoodSales))) \\
& \quad \cup (\pi_{clerk}(\overline{Emp}) \setminus \\
& \quad\ \pi_{clerk}(\sigma_{price>10000}(GoodSales))) \\
\approx\ & (\pi_{clerk}(C_{Emp}) \setminus \\
& \quad\ \pi_{clerk}(\sigma_{price>10000}(GoodSales))) \\
& \quad \cup (\pi_{clerk}(\pi_{attr(Emp)}(YoungSales)) \setminus \\
& \quad\ \pi_{clerk}(\sigma_{price>10000}(GoodSales)))
\end{aligned}$$

Therefore, we obtain

$$\begin{aligned}
\overline{op_3} =_{df}\ & (\pi_{clerk}(C_{Emp}) \setminus \\
& \quad\ \pi_{clerk}(\sigma_{price>10000}(GoodSales))) \\
& \quad \cup (\pi_{clerk}(\pi_{attr(Emp)}(YoungSales)) \setminus \\
& \quad\ \pi_{clerk}(\sigma_{price>10000}(GoodSales))),
\end{aligned}$$

which shows that $C_{Emp}$ is necessary to rewrite $op_3$ over $V \cup C$. As a consequence, we have $A_{op_3} = \{C_{Emp}\}$.

Concerning $op_4 =_{df}$ insert $\Delta e$ into $Emp$ we are now going to derive the translated update $\overline{op_4}$ that maintains the warehouse views in response to $op_4$. We note that $Emp$ occurs in the warehouse view $YoungSales$, but not in $ItemDesc$ or $GoodSales$; moreover, $Emp$ occurs in the complementary views $C_{Emp}$ and $C_{Sale}$, but not in $C_{Catalog}$. Therefore, $YoungSales$, $C_{Emp}$, and $C_{Sale}$ are the only views that could be affected by $op_4$. Consequently, we proceed to derive maintenance expressions for these views. Let $d$ be a state of $D$, and let $d'$ be the new state after execution of $op_4$. Then we have:

$$\begin{aligned}
& YoungSales(d') \\
& = \sigma_{age<30}(Sale \bowtie Emp)(d') \\
& = \sigma_{age<30}(Sale(d') \bowtie Emp(d')) \\
& = \sigma_{age<30}(Sale(d) \bowtie (Emp(d) \cup \Delta e)) \\
& = \sigma_{age<30}(Sale(d) \bowtie Emp(d)) \\
& \quad \cup \sigma_{age<30}(Sale(d) \bowtie \Delta e) \\
& = YoungSales(d) \cup Sale(d) \bowtie \sigma_{age<30}(\Delta e) \\
& = YoungSales(d)
\end{aligned}$$

We note that the last of the above equalities holds due to the foreign key between $Sale$ and $Emp$: no newly inserted tuple in $Emp$ can join with a previously existing tuple in $Sale$; hence, $\sigma_{age<30}(Sale(d) \bowtie \Delta e)$ is always empty. Consequently, the state of $YoungSales$ is not affected by $op_4$; moreover, as no complementary view is necessary to maintain the views in $V$, we obtain $A_{op_4} = \emptyset$.

Since we know that $YoungSales$ and $GoodSales$ are not affected by $op_4$, it is easy to see that $C_{Sale} \approx Sale \setminus (\pi_{attr(Sale)}(YoungSales) \cup \pi_{attr(Sale)}(GoodSales))$ is not affected by $op_4$ either.

The new state for $C_{Emp}$ in response to $op_4$ is now computed as follows:

$$\begin{aligned}
& C_{Emp}(d') \\
& = (Emp \setminus \overline{Emp})(d') \\
& = Emp(d') \setminus \overline{Emp}(d') \\
& = (Emp(d) \cup \Delta e) \setminus \overline{Emp}(d') \\
& = (Emp(d) \cup \Delta e) \setminus \pi_{attr(Emp)}(YoungSales)(d') \\
& = (Emp(d) \cup \Delta e) \setminus \pi_{attr(Emp)}(YoungSales)(d) \\
& = (Emp(d) \setminus \pi_{attr(Emp)}(YoungSales)(d)) \\
& \quad \cup (\Delta e \setminus \pi_{attr(Emp)}(YoungSales)(d)) \\
& = C_{Emp}(d) \cup (\Delta e \setminus \pi_{attr(Emp)}(YoungSales)(d)) \\
& = C_{Emp}(d) \cup \Delta e
\end{aligned}$$

We note that the last of the above equalities holds due to the key constraint on $Emp$ and the foreign key between $Emp$ and $Sale$: these constraints imply that no newly inserted tuple in $\Delta e$ can already be contained in the join involved in $YoungSales$.

To summarize, the only view in $W$ that needs to be maintained in response to $op_4$ is $C_{Emp}$. Thus, $\overline{op_4}$ is defined by $C_{Emp}(d') = C_{Emp}(d) \cup \Delta e$.

Concerning $op_5 =_{df}$ insert $\Delta s$ into $Sale$ we note that $Sale$ occurs in the warehouse views $GoodSales$ and $YoungSales$ as well as in all complementary views. Therefore, $\overline{op_5}$ is defined by maintenance expressions for these views, which we derive next. Let $d$ be a state of $D$, and let $d'$ be the new state after execution of $op_5$. Similar computations as above give the following:

$$\begin{aligned} &GoodSales(d') \\ &= GoodSales(d) \cup \pi_{attr(GoodSales)}(\sigma_{price>1000}( \\ &\quad \Delta s \bowtie (C_{Catalog}(d) \cup Catalog^{llj}(d)))) \end{aligned} \tag{5}$$

$$\begin{aligned} &YoungSales(d') \\ &= YoungSales(d) \cup \sigma_{age<30}( \\ &\quad \Delta s \bowtie (C_{Emp}(d) \cup \overline{Emp}(d))) \end{aligned} \tag{6}$$

The above equations show that $C_{Catalog}$ is necessary to determine the new state of $GoodSales$ and that $C_{Emp}$ is necessary for $YoungSales$. Consequently, we obtain $A_{op_5} = \{C_{Catalog}, C_{Emp}\}$.

Concerning the maintenance expressions for $C$, we have:

$$\begin{aligned} &C_{Emp}(d') \\ &= C_{Emp}(d) \setminus \pi_{attr(Emp)}(\sigma_{age<30}( \\ &\quad \Delta s \bowtie (C_{Emp}(d) \cup \overline{Emp}(d)))) \end{aligned} \tag{7}$$

$$\begin{aligned} &C_{Catalog}(d') \\ &= Catalog(d) \setminus (\pi_{attr(Catalog)}((GoodSales(d) \\ &\quad \cup \pi_{attr(GoodSales)}(\sigma_{price>1000} \\ &\quad\quad (\Delta s \bowtie (C_{Catalog}(d) \cup Catalog^{llj}(d))))) \\ &\quad \bowtie ItemDesc(d))) \end{aligned} \tag{8}$$

$$\begin{aligned} &C_{Sale}(d') \\ &= (Sale(d) \cup \Delta s) \setminus (\pi_{attr(Sale)}(GoodSales(d) \\ &\quad \cup \pi_{attr(GoodSales)}(\sigma_{price>1000}( \\ &\quad\quad \Delta s \bowtie (C_{Catalog}(d) \cup Catalog^{llj}(d))))) \\ &\quad \cup \pi_{attr(Sale)}(YoungSales(d) \\ &\quad \cup \sigma_{age<30}(\Delta s \bowtie (C_{Emp}(d) \cup \overline{Emp}(d))))) \end{aligned} \tag{9}$$

To summarize, $\overline{op_5}$ is defined by insertions into $GoodSales$ and $YoungSales$ as stated by Eqs. (5) and (6) above, respectively, and updates for $C_{Emp}$, $C_{Catalog}$, and $C_{Sale}$ as stated by Eqs. (7), (8), and (9) above, respectively.

*Step 3.* The results of Step 2 imply $A_{op_1} = \emptyset$, $A_{op_2} = \{C_{Catalog}\}$, $A_{op_3} = \{C_{Emp}\}$, $A_{op_4} = \emptyset$, and $A_{op_5} = \{C_{Emp}, C_{Catalog}\}$. Thus, Theorem 3.3 implies that $V \cup A_{OP}$ is an $OP$-independent warehouse, where $A_{OP} = \{C_{Emp}, C_{Catalog}\}$. In particular, no auxiliary information concerning base relation $Sale$ is necessary. Finally, it is easy to see that $A_{op_i}$ is a unique minimal subset of $C$ that ensures $op_i$ independence, $i = 1, \ldots, 5$. Hence, Theorem 3.4 asserts that $A_{OP}$ is the unique minimal subset of $C$ that ensures $OP$-independence.

Finally, we emphasize that modifications can be handled by the independence steps as well. Consider the following modification:

$$op_6 =_{df} \text{update } Sale \text{ set amount = } x \text{ where } y$$

We recall that the warehouse under consideration is not update-independent with respect to insertions into $Sale$ (as the independence steps produce $A_{op_5} = \{C_{Emp}, C_{Catalog}\}$ for the insertion into $Sale$ specified by $op_5$). Nevertheless, the warehouse is $op_6$-independent, as we will indicate next.

Let $d$ be a state of $D$, and let $d'$ be the new state after execution of $op_6$. Since $op_6$ is a modification to $Sale$, we have $Sale(d') = (Sale(d) \cup \Delta^+ s) \setminus \Delta^- s$, where every item that occurs in $\Delta^+ s$ or $\Delta^- s$ must also occur (with a different amount) in $Sale(d)$.

To simplify notation let $G = attr(GoodSales)$ and $\phi = price > 1000$. Then we have:

$$\begin{aligned} &GoodSales(d') = \pi_G(\sigma_\phi(Sale(d') \bowtie Catalog(d'))) \\ &= \pi_G(\sigma_\phi(((Sale(d) \cup \Delta^+ s) \setminus \Delta^- s) \bowtie Catalog(d))) \\ &= GoodSales(d) \cup \pi_G(\sigma_\phi(\Delta^+ s \bowtie Catalog(d))) \\ &\quad \setminus \pi_G(\sigma_\phi(\Delta^- s \bowtie Catalog(d))) \end{aligned}$$

Now, since every item that occurs in $\Delta^+ s$ or $\Delta^- s$ also occurs in $Sale(d)$, we know that all tuples in $\Delta^+ s$ and $\Delta^- s$ that contribute to $GoodSales(d')$ have join partners in $Catalog(d)$. Moreover, $GoodSales(d)$ contains the key $item$ for all join partners that satisfy the selection condition $price > 1000$. Therefore, using $SoldItem =_{df} \pi_{item,price}(GoodSales)$ we obtain

$$\begin{aligned} GoodSales(d') = GoodSales(d) \\ \cup \pi_G(\sigma_\phi(\Delta^+ s \bowtie SoldItem(d))) \\ \setminus \pi_G(\sigma_\phi(\Delta^- s \bowtie SoldItem(d))), \end{aligned}$$

which can be computed from warehouse views and update information. Consequently, no complementary information is necessary to maintain $GoodSales$. Similar computations apply to view $YoungSales$, which can be maintained without complementary views as well, and $Sale$ does not occur in $ItemDesc$, which shows that the warehouse is $op_6$-independent.

We end this section with some remarks concerning the cost of performing the independence steps. First, all computations involve schema manipulations only and require no access to data whatsoever. Second, the independence steps require some skill in the manipulation of relational expressions. However, we would like to emphasize in this respect that performing the independence steps is a one-shot operation, required only at the design phase, and performed by the data warehouse designer who presumably does have such skills.

Moreover, we note that throughout the above examples we have given canonical rewritings in our context. In general, the problem of finding such rewritings relates to "answering queries using views" ([8,20,26]) a very difficult problem in general. Clearly, whatever optimization results exist or will exist in that area can be used in our case as well. However, this topic in itself, lies outside the scope of the present paper.

## 4 Related work

We next review work that is related to ours. The notion of a warehouse complement we use here derives directly from the notion of view complement first introduced in [4]. However, view complements were used in [4] to translate updates on the view back to updates on the underlying database. Here we use complements to translate in the opposite direction, i.e., to translate queries and updates on the database to queries and updates on the view (i.e., on the warehouse).

The computation of complements for views defined by relational algebra was first discussed in [9]. The approach of [9] 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 [9] states that even in this simple setting, finding a minimal complement (where a "minimal" complement is a projection with as few attributes as possible) is NP-complete. This result does not carry over to our setting, since: (a) we do not consider arbitrary functional dependencies; (b) our complements and inverses have a different form; and (c) our notion of minimality is completely different. On the other hand, our approach allows to compute complements for *sets* of views defined by projection, selection, and join, and the results of Sect. 2.3.3 show that the computation of complementary views is, roughly, exponential in the number of warehouse views.

The present paper is an extension of [19], where we have shown how to compute complements for sets of views defined by projection, selection, and join in the presence of key and inclusion dependencies. Moreover in [19], we have defined the notion of warehouse independence, and we have shown that independence with respect to all queries and all updates can be achieved by adding a complement to a warehouse. In this paper we improve over [19] in the following important and nontrivial points:

1. We define and study independence for *arbitrary* sets of queries and updates.
2. We provide independence steps to compute a *minimal* set of auxiliary views for rendering the warehouse independent.

Concerning the complements computed here and in [19], we finally remark that the approach presented in [3] shows how to exploit these complements to make temporal views over non-temporal sources self-maintainable.

We next clarify some terminology concerning self-maintainability and independence. The idea of maintaining views without looking at the underlying base relations goes back at least to [5]. The problems addressed in [5] are the following ones. Consider a single materialized view $V$ over base relations $D$. Given a base relation update, determine conditions under which the update cannot affect the current instance of $V$, and determine conditions under which the effects of the update on the current instance of $V$ can be computed without base relation access. Updates of the former kind are called *irrelevant updates* while updates of the latter kind are called *autonomously computable* in [5]. The main results of [5] are necessary and sufficient conditions characterizing both kinds of updates for a given view $V$, which is defined using the relational operations selection, projection, and join.

We note that an irrelevant update is always autonomously computable, but the converse is clearly not true. Moreover, it is easy to see that a view is $u$-independent for an update $u$ if and only if $u$ is autonomously computable on $V$ in the sense of [5]. Hence, the work of [5] can be seen as a precursor for research on self-maintainability, which has been identified as a desirable warehouse property in [12]. The work of [17] presents algorithms that decide whether a given set of views is self-maintainable or not.

Furthermore, we observe that irrelevant updates are studied in detail in [21]. In [21] a query (given as datalog program) is called *independent* of an update (specified as another datalog program), if the update does not change the answer to the query, i.e., if the update is irrelevant in the sense of [5]. Independence is reduced to the equivalence problem for datalog programs, and query-reachability and uniform equivalence are identified as sub-classes of equivalence, which provide decidable conditions for independence.

In the warehousing context, the results of [21] could be exploited in a filtering step at the source layer, which detects irrelevant updates that cannot have any effect on the warehouse state and that consequently do not need to be sent to the warehouse. Only those updates that can possibly affect the warehouse state need then to be shipped to and integrated into the warehouse. Clearly, in such a scenario it is still desirable to make the data warehouse self-maintainable.

The first technique to *make* a given warehouse self-maintainable by using "auxiliary" views seems to be the one presented in [25]. The approach followed in [25] is to first determine a set $\{E_1, \ldots, E_n\}$ of so-called *maintenance expressions* (assuming a single materialized view $V$) and then to proceed in either of two different ways:

1. From the maintenance expressions $E_1, \ldots, E_n$, extract auxiliary views that, together with the warehouse view, are self-maintainable with respect to updates.
2. Given the warehouse view $V$, "guess" a set of auxiliary views $\{A_1, \ldots, A_l\}$ and then
   - check if the maintenance expressions $E_1, \ldots, E_n$ can be computed from $W = \{V, A_1, \ldots, A_l\}$, using an algorithm to determine whether a query can be answered using a set of views [8,20,26] and
   - check if the views $\{A_1, \ldots, A_l\}$ can be maintained from $W$.

The work reported in [2] is an extension of [25] towards the self-maintainability of a single *generalized PSJ* view, i.e., a PSJ expression, where the projection may include group-by and aggregate attributes.

In fact, our approach is the "opposite" of that of [2,25], in the sense that we first determine the auxiliary views and then compute the maintenance expressions. In doing so, we assume *any* number of materialized views – not just a single view as in [2,25].

A different approach towards self-maintainability of a single PSJ view is proposed in [29]: here, data sources are represented using tables with variables, whereas the warehouse as well as auxiliary views are modeled as conditional tables. Then updates are perceived as assignments of values to variables, which allows to compute new warehouse states using the conditional (warehouse) tables relative to new variable assignments. In contrast to our work, the results of [29] are only applicable in warehouse environments, where source relations may contain variables and the warehouse itself is a conditional database.

Before we continue to present further related work, we would like to recall that a PSJ view is, in general, not self-maintainable with respect to deletions [12]. A sufficient condition for self-maintainability (with respect to deletions) of PSJ views is that the view preserves a key of each base relation occurring in the view definition [12]. Indeed, this observation is already true for views involving projections only: E.g., given a relation name $R$ with $attr(R) = \{A, B\}$ and a view $V =_{df} \pi_A(R)$, it is easy to see that $V$ is only self-maintainable, if $A$ is the key of $R$.

In fact, the algorithm for self-maintainability of [25] derives one auxiliary view per base relation, and the projection involved in this view includes the key. However, this algorithm *always* produces one non-empty auxiliary view per base relation – even if the original view is self-maintainable without further information. Thus, the claim of [25] that the set of auxiliary views produced by this algorithm is minimal (with respect to set inclusion), is clearly false.

Next, [22] claims to be an extension of [25] towards self-maintainability of *sets* of PSJ views. However, this approach does not consider keys and foreign keys; hence, it falls short of including a key into the list of projected attributes in auxiliary views. As a consequence, the algorithm presented in [22] fails to make views of the form $\pi_X(R)$ self-maintainable, when $X$ does not include a key of $R$. Moreover (similarly to [25]), the algorithm produces auxiliary information, even if the original set of views is already self-maintainable.

The approach described in [23] considers the problem of deriving auxiliary views to ensure self-maintainability of a single view that is defined using relational algebra with aggregation. In this approach, a view is represented using an expression tree, where: (a) the base relations are leafs; (b) projections and selections are associated to edges; and (c) binary operations as well as group-by operations are inner nodes. Based on this representation, the authors compute auxiliary views in such a way that a view maintenance algorithm can determine the "exact" changes to each subexpression of the view bottom-up in the expression tree.

The approach of [23] should be improved with respect to the following points: first, similarly to [22], the proposed view maintenance algorithm is based on the assumption that changes for subexpressions of the form $\pi_X(\sigma_\phi(R))$ are always available, which is not true in general. Second, in order to determine the auxiliary relations, the authors assume that key constraints are given for the *inner nodes* of an expression tree. However, [23] does not contain any hint of how these keys could be obtained. Finally, concerning self-maintainability of a join of two base relations, according to [23] both base relations are materialized at the warehouse. However, our approach shows that the complement is: (a) sufficient for update *and* query independence; and (b) strictly smaller than the base relations (consider the auxiliary views defined in Fig. 1).

## 5 Concluding remarks and future work

We have presented an approach for specifying independent warehouses, i.e., warehouses that are independent of data sources in answering source queries *and* in processing source updates. The key idea behind our approach is setting up a one-to-one mapping from database states to warehouse states, so that the warehouse can compute all base relations, if necessary. We have seen that this idea can be implemented by adding a complement to the warehouse, and we have given an algorithmic approach for computing monotonic complements for warehouses defined by PSJ expressions from databases containing keys and foreign keys.

We remark that the complementary relations $\{C_{R_1}, \ldots, C_{R_n}\}$ given by Proposition 2.3, or the subset thereof produced by the independence steps, are *all* the information we need for warehouse independence. If the queries to base relations required for the computation of any specific $C_{R_i}$ can be answered in reasonable time, then we do not need to maintain $C_{R_i}$ at the warehouse; we simply store the expression for computing it. Otherwise, we have to maintain $C_{R_i}$ at the warehouse.

Next, it is current practice to build warehousing environments on top of star schemes around fact and dimension tables which integrate information from various sources [7,28]. If we assume all sources to be relational, then each fact table can be regarded as a PSJ view (or unions thereof) and maintained using our approach. For example, consider a business warehouse where parts from different suppliers are sold to customers according to their orders (similar to the one modeled in the TPC-H decision support benchmark [30]). This business could be distributed over several locations, each running its own operational database. Now, the warehouse maintains:

a) dimension tables to store data on locations, customer, and supplier and;
b) fact tables (including foreign keys from the dimension tables) for orders and sales which are extracted by PSJ queries from the sources and integrated by union.

Although views including union cannot be used for computing complements in general, the presence of foreign keys allows us to uniquely determine the origin of each tuple in a fact table by selecting on the dimension attributes. Thus, we can even exploit fact tables, that are integrated by union, for computing the warehouse complement. As a result, star schemes allow for an even wider applicability of our approach.

Furthermore, as mentioned in the Introduction, OLAP is a major application domain for data warehousing, where analysts execute complex queries involving aggregate views defined on fact tables. Although aggregate queries cannot be exploited when computing complements, they do *not* restrict the applicability of our approach either: the fact tables can be maintained as described above using PSJ views, whereas view maintenance algorithms for aggregate queries, e.g., [10,15, 24], can be used to maintain materialized aggregate queries.

Several lines of future research are envisaged. The computation of minimal complements needs to be studied in more

depth. In this paper we have assumed that each view in the complement has the same set of attributes as some base relation. Relaxing this restriction may lead to smaller complements. Moreover, the computation of complements for a larger class of views is still an open problem. However, Example 3.4 raises the question for which extensions of views and complements the uniqueness and minimality results of Theorem 3.2 and Theorem 3.4 still hold.

Finally, in the present paper we have studied the concepts involved in warehouse independence, and we have proposed the independence steps as an algorithmic approach to establish independence. A single algorithm, however, which takes as input: (a) a warehouse $V$; and (b) a set of operations $OP$, and which produces as output: (a) a set of auxiliary views $A$ such that $V \cup A$ is $OP$-independent; and (b) a set of translated warehouse operations, needs still to be devised.

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

## References

1. Abiteboul S., Hull R., Vianu V. Foundations of databases. Addison-Wesley, Reading, Mass., USA, 1995
2. Akinde M.O., Jensen O.G., Böhlen M.H. Minimizing detail data in data warehouses. Proc. 6th EDBT 1998, pp 293–307
3. de Amo S., Halfeld Ferrari Alves M. Efficient maintenance of temporal data warehouses. Proc. IDEAS 2000, pp 188–196
4. Bancilhon F., Spyratos N. Update semantics of relational views. ACM TODS 6:557–575, 1981
5. Blakeley J.A., Coburn N., Larson P. Updating derived relations: detecting irrelevant and autonomously computable updates. ACM TODS 14:369–400, 1989
6. Blakeley J.A., Larson P., Tompa F.W. Efficiently updating materialized views. Proc. ACM SIGMOD 1986, pp 61–71
7. Chaudhuri S., Dayal U. An overview of data warehousing and OLAP technologies. Proc. ACM SIGMOD 1997, pp 65–74
8. Chaudhuri S., Krishnamurthy R., Potamianos S., Shim K. Optimizing queries with materialized views. Proc. 11th ICDE 1995, pp 190–200
9. Cosmadakis S.S., Papadimitriou C.H. Updates of relational views. JACM 31:742–760, 1984
10. Griffin T., Libkin L. Incremental maintenance of views with duplicates. Proc. ACM SIGMOD 1995, pp 328–339
11. Griffin T., Libkin L., Trickey H. An improved algorithm for the incremental recomputation of active relational expressions. IEEE TKDE 9:508–511, 1997
12. Gupta A., Jagadish H.V., Mumick I.S. Data integration using self-maintainable views. Technical Memorandum, AT&T Bell Laboratories, November 1994
13. Gupta A., Mumick I.S. Maintenance of materialized views: problems, techniques, and applications. IEEE Data Eng Bull 18(2):3–18, 1995
14. Gupta A., Mumick I.S. (eds) Materialized views: techniques, implementations, and applications. The MIT Press, Cambridge, Mass., USA, 1999
15. Gupta A., Mumick I.S., Subrahmanian V.S. Maintaining views incrementally. Proc. ACM SIGMOD 1993, pp 157–167
16. Honeyman P. Extension joins. Proc. 6th VLDB 1980, pp 239–244
17. Huyn N. Multiple-view self-maintenance in data warehousing environments. Proc. 23rd VLDB 1997, pp 26–35
18. Karp R.M. Reducibility among combinatorial problems. In: Complexity of computer computations. Plenum, New York, 1972, pp 85–103
19. Laurent D., Lechtenbörger J., Spyratos N., Vossen G. Complements for data warehouses. Proc. 15th ICDE 1999, pp 490–499
20. Levy A.Y., Mendelzon A., Sagiv Y., Srivastava D. Answering queries using views. Proc. 14th ACM PODS 1995, pp 95–104
21. Levy A.Y., Sagiv Y. Queries independent of updates. Proc. 19th VLDB 1993, pp 171–181
22. Liang W., Li H., Wang H., Orlowska M.E. Making multiple views self-maintainable in a data warehouse. DKE 30:121–134, 1999
23. Mohania M.K., Kambayashi Y. Making aggregate views self-maintainable. DKE 32:87–109, 2000
24. Mumick I.S., Quass D., Mumick B.S. Maintenance of data cubes and summary tables in a warehouse. Proc. ACM SIGMOD 1997, pp 100–111
25. Quass D., Gupta A., Mumick I.S., Widom J. Making views self-maintainable for data warehousing. Proc. PDIS 1996
26. Rajaraman A., Sagiv Y., Ullman J.D. Answering queries using templates with binding patterns. Proc. 14th ACM PODS 1995, pp 95–104
27. Sagiv Y., Yannakakis M. Equivalences among relational expressions with the union and difference operators. JACM 27:633–655, 1980
28. Sen A., Jacob V.S. (eds) Industrial-strength data warehousing. CACM 41(9):28–69 1998
29. Shu H. View maintenance using conditional tables. Proc. 5th DOOD 1997, Lecture Notes in Computer Sciance, vol. 1341. Springer, Berlin Heidelberg New York, 1997, pp. 67–84
30. Transaction Processing Council. TPC Benchmark(tm) H (Decision Support), Standard Specification, Revision 1.2.1. San Jose, 1999. Available on the web under URL http:// www.tpc.org/tpch
31. Ullman J.D. Principles of database and knowledge-base systems, vol I. Computer Science, New York, 1988
32. Vossen G. Data models, database languages, and database management systems. Addison-Wesley, Reading, Mass., USA, 1991
33. Widom J. (ed) Special issue on materialized views and data warehousing. IEEE Data Eng Bull 18(2), 1995
34. Widom J. Research problems in data warehousing. Proc. 4th CIKM 1995
35. Zhuge Y., Garcia-Molina H., Hammer J., Widom J. View maintenance in a warehousing environment. Proc. ACM SIGMOD 1995, pp 316–327
36. Zhuge Y., Garcia-Molina H., Wiener J.L. Multiple view consistency for data warehousing. Proc. 13th ICDE 1997, pp 289–300

## A Appendix

### *A.1 Proof of Theorem 2.1*

**Theorem 2.1.** Let $V = \{V_1, \ldots, V_k\}$ be a set of SJ views over $D$. Then the canonical complement of $V$ is a minimal complement.

*Proof.* Let $C = \{C_{R_1}, \ldots, C_{R_n}\}$ be the canonical complement of $V$, and consider a set of views $C' = \{C'_{R_1}, \ldots, C'_{R_n}\}$ such that $C' < C$, i.e., we have $C'_{R_i}(d) \subseteq C_{R_i}(d)$ for all states $d$ and $i \in [1, n]$, and there is a state $d_0 = \langle r_1, \ldots, r_n \rangle$ such that $C'_{R_i}(d_0) \subsetneq C_{R_i}(d_0)$ for some $i \in [1, n]$. For the sake of readability, we assume $i = 1$ and $C'_{R_j} \approx C_{R_j}$ for $j \in [2, n]$.

(The proof can easily be adapted to the case where $C'$ contains multiple complementary views that are strictly smaller than their counterparts in $C$.) We proceed by showing that $C'$ is not a complement of $V$ with respect to $D$.

Consider the sequence of states $(d_j)_{j\geq 0}$, where $d_j = \langle C'_{R_1}(d_{j-1}) \cup \overline{R_1}(d_0), r_2, \ldots, r_n\rangle$ for $j \geq 1$. Let $d_r$ be the representative state for $V$ and $d_0$. We observe that $R(d_r) \subseteq R(d_j) \subseteq R(d_0)$. Hence, Lemma 2.1 (3) implies $V(d_j) = V(d_0)$ for $j \geq 0$.

Moreover, the sequence $r_1^{(j)}$ of relations over $R_1$ defined by $r_1^{(j)} = R_1(d_j)$ is monotonically decreasing: Recall that for $j \geq 0$ we have $V(d_j) = V(d_0)$. Hence, we have $C_{R_1}(d_j) = C'_{R_1}(d_{j-1})$. As we assume $C' < C$, we find $C'_{R_1}(d_j) \subseteq C_{R_1}(d_j) = C'_{R_1}(d_{j-1})$, i.e., $C'_{R_1}(d_j) \subseteq C'_{R_1}(d_{j-1})$.

By monotonicity, $r_1^{(j)}$ has a limit. This implies the existence of some $l > 0$ such that $r_1^{(l+1)} = r_1^{(l)} \subsetneq r_1^{(l-1)}$, which in turn results in $C'_{R_1}(d_l) = C'_{R_1}(d_{l-1})$ where $d_l \neq d_{l-1}$. Hence, we have $\langle C'(d_l), V(d_l)\rangle = \langle C'(d_{l-1}), V(d_{l-1})\rangle$, and by Proposition 2.2, this implies that $d_l = d_{l-1}$; a contradiction showing that $C'$ is not a complement of $V$ with respect to $D$. □

### A.2 Proof of Theorem 3.1

**Theorem 3.1.** Let $V$ be a warehouse over $D$, and let $C$ be a monotonic complement of $V$. Let $op$ be an operation over $D$, and let $A_{op}$ be the set of auxiliary views produced according to the independence steps. Then $W = V \cup A_{op}$ is an $op$-independent warehouse.

*Proof.* If $op$ is a query, then $op$-independence follows immediately from Definition 3.2 and from the construction of $A_{op}$.

If $op$ is an update, then it is easy to see that the independence steps determine a subset $A_{op}$ of $C$ such that the new state of $V$ after $u$ can be derived from $V \cup A_{op}$. It remains to show that the new state of every complementary view in $A_{op}$ can be derived from $V \cup A_{op}$.

Let $C_{R_0} \in A_{op}$. As $C$ is monotonic, for every database state $d = \langle r_1, \ldots, r_0, \ldots, r_n\rangle$, we have $C_{R_0}(d) = (R_0 \setminus E_0)(d) = R_0(d) \setminus E_0(d)$. Let $\overline{u}$ be the translation of $u$ in terms of the views in $V \cup A_{op}$, and let $d' = \langle r'_1, \ldots, r'_0, \ldots, r'_n\rangle$ be the database state $u(d)$. Thus we have $C_{R_0}(d') = R_0(d') \setminus E_0(d')$. We now show that the subexpressions $R_0(d')$ and $E_0(d')$ in the computation of $C_{R_0}(d')$ can be computed using only information from $V \cup A_{op}$.

1. Let $\mathcal{E}_0$ be the inverse expression of the monotonic complementary view $C_{R_0}$ for $R_0$. Then, based on Definition 2.4, $\mathcal{E}_0$ is defined over $E_0$ and $C_{R_0}$. Thus, the presence of $C_{R_0}$ allows to compute $R_0(d)$ from warehouse views using the expression $R_0(d) = \mathcal{E}_0(E_0(d), C_{R_0}(d))$. Moreover, as the changes to $R_0(d)$ are shipped to the warehouse, the new state of $R_0$, $R_0(d')$, can be computed at the warehouse.

2. On the other hand, $E_0$ is an expression where only views from $V$ occur, $V \cup A_{op}$ is a superset of $V$, and $V \cup A_{op}$ is determined by the independence steps in such a way that $V(d')$ can be computed from $V \cup A_{op}$. As a consequence, the new state for each view in $V$ can be computed at the warehouse. Evaluating $E_0$ on this new view state then gives $E_0(d')$.

Consequently, the new state for $C_{R_0}$, $C_{R_0}(d') = R_0(d') \setminus E_0(d')$, can be computed from $V \cup A_{op}$, which concludes the proof. □

### A.3 Proof of Theorem 3.2

**Theorem 3.2.** Let $V$ be a warehouse over $D$, let $C$ be a monotonic complement of $V$, and let $op$ be an operation over $D$.

1. Then there is a subset $A_{op}$ of $C$ such that
   (a) $V \cup A_{op}$ is $op$-independent and
   (b) $A_{op}$ is minimal with respect to set inclusion, i.e., if we remove any of the views in $A_{op}$ then (a) above does not hold.
2. If there are no foreign keys declared over $D$ then the set $A_{op}$ mentioned in (1) is unique.

*Proof.* Existence of $A_{op}$ is not an issue, as $C$ itself ensures independence. Moreover, by considering each subset of $C$ that ensures $op$-independence a minimal one can be identified.

Concerning uniqueness, let $A_{op}$ be a minimal subset of $C$ that ensures $op$-independence, and let $A'_{op}$ be a subset of $C$ that does not include all views of $A_{op}$, say $C_{R_1} \in A_{op} \setminus A'_{op}$. As $C$ is a monotonic complement, $C_{R_1}$ is the complementary view for base relation $R_1$. The fact that $C_{R_1}$ is contained in $A_{op}$, which is a minimal subset of a monotonic complement that guarantees $op$-independence, tells us that some information from $R_1$ that is necessary for $op$-independence is missing in the views $V$. Formally (using the notion of information content of [4]), this means that there are at least two legal states of $D$ (i.e., states that satisfy all key constraints) that are distinguished by $op$ and $V \cup A_{op}$, but not by $V \cup A_{op} \setminus \{C_{R_1}\}$. Let $d_1 = \langle r_1^1, \ldots, r_n^1\rangle$ and $d_2 = \langle r_1^2, \ldots, r_n^2\rangle$ be witnesses for this fact, i.e., we have

(a) $V(d_1) = V(d_2)$,
(b) $(A_{op} \setminus \{C_{R_1}\})(d_1) = (A_{op} \setminus \{C_{R_1}\})(d_2)$,
(c) $C_{R_1}(d_1) \neq C_{R_1}(d_2)$
(this condition must hold as $V$ and $A_{op}$ together are supposed to distinguish $d_1$ and $d_2$, and $C_{R_1}$ is the only view in $V \cup A_{op}$ that is not excluded by (a) and (b)),
(d) $op(d_1) = \overline{op}((V \cup A_{op})(d_1)) \neq \overline{op}((V \cup A_{op})(d_2)) = op(d_2)$
(if $op$ would not distinguish $d_1$ and $d_2$ then no additional information for $op$-independence would be necessary to tell both states apart).

For the purposes of this proof, we assume that PSJ expressions always are of the form $\pi_Z(\sigma_\phi(R_{i_1} \bowtie \ldots \bowtie R_{i_k}))$, where $\phi \equiv true$ if no selection is necessary, and where $Z = \bigcup_{j=1}^{k} attr(R_{i_j})$ if no projection is necessary.

Let $V^{SJ}$ be the set of SJ views over $D$ that is obtained from $V$ by removing the projection from each view, i.e., $V^{SJ} = \{\sigma_\phi(R_{i_1} \bowtie \ldots \bowtie R_{i_k}) \mid \pi_Z(\sigma_\phi(R_{i_1} \bowtie \ldots \bowtie R_{i_k})) \in V\}$. In the following we use the representative state for $V^{SJ}$ and $d_1$ to obtain two legal states of $D$ showing that $A'_{op}$ does not contain enough information for $op$-independence.

Let $d_r$ be the representative state for $V^{SJ}$ and $d_1$. Then, by Lemma 2.1 (1) we have $R(d_r) \subseteq R(d_1)$ for all $R \in D$. Since $d_1$ is supposed to be a legal state of $D$, all key constraints are valid in $d_r$ (if a relation satisfies a key constraint then so does every subset), which proves that $d_r$ is a legal state of $D$.

Next, by Lemma 2.1 (2) we have $V^{SJ}(d_r) = V^{SJ}(d_1)$. Hence, by applying the original projections to the views in $V^{SJ}$, we find $V(d_r) = V(d_1)$ ($= V(d_2)$ by (a) above).

Consider the states $d'_1$ and $d'_2$ of $D$ that are defined as follows for $R \in D$, $i = 1, 2$:

$$R(d'_i) = \begin{cases} R(d_i) \text{ if } R \in \{R \in D \mid C_R \in A_{op}\} \\ R(d_r) \text{ otherwise} \end{cases}$$

As the relation states associated with $d'_1$ and $d'_2$ do either occur in $d_r$, in $d_1$, or in $d_2$, all of which are legal states of $D$, all key constraints are satisfied in $d'_1$ and $d'_2$, i.e., $d'_1$ and $d'_2$ are legal states of $D$ as well.

We now proceed to establish several facts, which are then used to prove that $A'_{op}$ does not guarantee $op$-independence.

1. $R_1(d'_1) \neq R_1(d'_2)$
   Recall that $C$ contains a monotonic complementary view $C_{R_1} =_{df} R_1 \setminus E_1$ for $R_1$, where $E_1$ is an expression over $V$. By (a) we have $V(d_1) = V(d_2)$, which implies $E_1(d_1) = E_1(d_2)$. Then it follows from (c) that $R_1(d_1) \neq R_1(d_2)$. As we have $R_1(d_1) = R_1(d'_1)$ and $R_1(d_2) = R_1(d'_2)$, the claim follows.
2. $R(d_1) = R(d'_1) = R(d'_2) = R(d_2)$, for $R \in \{R \in D \mid C_R \in A_{op} \setminus \{C_{R_1}\}\}$
   Let $R \in \{R \in D \mid C_R \in A_{op} \setminus \{C_{R_1}\}\}$. Then (a) and (b) show that the (monotonic) inverse for $R$ yields the same result in $d_1$ and $d_2$, i.e., $R(d_1) = R(d_2)$. Using the definitions of $d'_1$ and $d'_2$, we obtain $R(d'_1) = R(d_1) = R(d_2) = R(d'_2)$.
3. $V(d'_1) = V(d_1) = V(d_2) = V(d'_2)$
   We observe $R(d_r) \subseteq R(d'_1) \subseteq R(d_1)$, for all $R \in D$. Thus, by Lemma 2.1, we have $V(d_1) = V(d'_1)$. The equality $V(d_2) = V(d'_2)$ is shown in the same way and the equality $V(d_1) = V(d_2)$ is (a).
4. $A_{op}(d_i) = A_{op}(d'_i), i = 1, 2$
   Facts 2 and 3 imply $(A_{op} \setminus \{C_{R_1}\})(d_i) = (A_{op} \setminus \{C_{R_1}\})(d'_i)$. As $d'_i$ is defined such that $R_1(d_i) = R_1(d'_i)$, the claim follows from fact 3.
5. $A'_{op}(d'_1) = A'_{op}(d'_2)$
   As (i) all views in $A'_{op}$ are associated to base relations whose states in $d'_1$ and in $d'_2$ coincide; and (ii) $V(d'_1) = V(d'_2)$ holds by fact 3, the claim follows.

On the one hand, facts 3, 4, and 5 above imply $V \cup A_{op}(d_i) = V \cup A_{op}(d'_i), i = 1, 2$; hence, using (d) above, we have $op(d'_1) \neq op(d'_2)$. On the other hand, by facts 3 and 6 above we have $(V \cup A'_{op})(d'_1) = (V \cup A'_{op})(d'_2)$, which implies $op(d'_1) = \overline{op}((V \cup A'_{op})(d'_1)) = \overline{op}((V \cup A'_{op})(d'_2)) = op(d'_2)$. Therefore, we have a contradiction showing that $A_{op}$ is unique. □