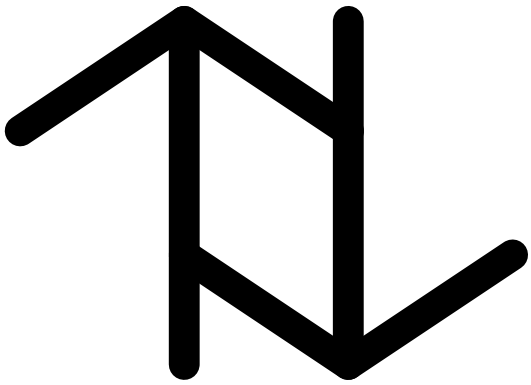

Persistent Views–A Mechanism for Managing Aging Data

Janne Skyt and Christian S. Jensen

TR-65

A TIMECENTER Technical Report

Title	Persistent Views–A Mechanism for Managing Aging Data
Author(s)	Janne Skyt and Christian S. Jensen
Publication History	September 2000. CoopIS (An extended abstract of this paper). July 2001. A TIMECENTER Technical Report.

TIMECENTER Participants

Aalborg University, Denmark
Christian S. Jensen (codirector), Michael H. Böhlen, Heidi Gregersen, Dieter Pfoser, Simonas Šaltenis, Janne Skyt, Giedrius Slivinskas, Kristian Torp

University of Arizona, USA
Richard T. Snodgrass (codirector), Dengfeng Gao, Vijay Khatri, Bongki Moon, Sudha Ram

Individual participants
Curtis E. Dyreson, Washington State University, USA
Fabio Grandi, University of Bologna, Italy
Nick Kline, Microsoft, USA
Gerhard Knolmayer, Universty of Bern, Switzerland
Thomas Myrach, Universty of Bern, Switzerland
Kwang W. Nam, Chungbuk National University, Korea
Mario A. Nascimento, University of Alberta, Canada
John F. Roddick, University of South Australia, Australia
Keun H. Ryu, Chungbuk National University, Korea
Michael D. Soo, amazon.com, USA
Andreas Steiner, TimeConsult, Switzerland
Vassilis Tsotras, University of California, Riverside, USA
Jef Wijsen, University of Mons-Hainaut, Belgium
Carlo Zaniolo, University of California, Los Angeles, USA

For additional information, see The TIMECENTER Homepage:
URL: `<http://www.cs.auc.dk/TimeCenter>`

The TIMECENTER icon on the cover combines two "arrows." These "arrows" are letters in the so-called *Rune* alphabet used one millennium ago by the Vikings, as well as by their precedessors and successors. The Rune alphabet (second phase) has 16 letters, all of which have angular shapes and lack horizontal lines because the primary storage medium was wood. Runes may also be found on jewelry, tools, and weapons and were perceived by many as having magic, hidden powers.

The two Rune arrows in the icon denote "T" and "C," respectively.

Abstract

Enabled by the continued advances in storage technologies, the amounts of on-line data grow at a rapidly increasing pace. This development is witnessed in, e.g., so-called data webhouses that accumulate click streams from portals, and in other data warehouse-type applications. The presence of very large and continuously growing amounts of data introduces new challenges, one of them being the need for effective management of aged data. In very large and growing databases, some data eventually becomes inaccurate or outdated, and may be of reduced interest to the database applications. This paper offers a mechanism, termed persistent views, that aids in flexibly reducing the volume of data, e.g., by enabling the replacement of such "low-interest," detailed data with aggregated data. The paper motivates persistent views and precisely defines and contrasts these with the related mechanisms of views, snapshots, and physical deletion. The paper also offers a provably correct foundation for implementing persistent views.

Key words and phrases: View, snapshot, vacuuming, logical deletion, physical deletion, temporal database, data warehousing.

1 Introduction

Data storage technologies continue to outperform Moore's Law and thus advance at a rapidly increasing pace. As a consequence, in his Turing Award lecture, Jim Gray predicted that there will be sold more data storage in the 18 months following his lecture than had been sold previously in all of history. Also, experience shows that data storage will be exploited to store increasing amounts of data as soon as it becomes available. The increasing amounts of data introduce new complexity in data management, offering new challenges.

This paper presents a new and flexible mechanism termed *persistent views*, or P-views for short, that helps managing such increasing amounts of data in append-only databases. Briefly, P-views are views that are immune to disciplined physical deletion, but function as regular views in the context of insertions, (logical) deletions, and updates. P-views offer the following benefits:

- They allow physical deletions to weed out data that no longer is desired while retaining continuously important information. For example, reasons for physically deleting data may be that data is outdated, inaccurate, no longer needed by any applications, or, that data must be deleted because it is traceable to specific individuals.
- They enable access to anonymous aggregate information based on detailed, possibly traceable, data that is to be physically deleted.
- They provide access control, eliminating access to base data.

The paper shows how P-views, in a flexible and user-friendly manner, enable the retention of, e.g., select, aggregate, or summary data, while also enabling the deletion of detailed data. When data is physically deleted from base relations on which P-views are defined, the base data that is necessary to compute the P-views and thus render them immune to the deletions is automatically and transparently extracted and retained. The paper offers a provably correct foundation for accomplishing this extraction and thus implementing P-views. (It should be noted that P-views are independent of the specific mechanism chosen for physical deletion.)

For illustration, consider click-stream data. Here, it may be desirable to retain only a high-granularity summary of web-usage data when this reaches a certain age. This summary data may be specified as one or several P-views, upon which the old, detailed access data may be physically deleted. The implementation of P-views ensures that detail data is reflected correctly in the summary data before it is physically deleted. This and another example application of P-views will be discussed more extensively in the next section.

As the context for P-views, the paper formalizes the append-only nature of many applications by introducing relations with transaction-time support [15]. In these applications, conventional deletion has only a logical effect, so a new mechanism is needed for physical deletion. To accomplish this, we employ vacuuming, which is a specific approach to physical deletion. The notion of vacuuming was most recently presented by Skyt et al. [11, 13]. P-views offer substantial benefits over vacuuming.

Next, views, i.e., named and stored query expressions, are fundamental in database management and have been the topic of a multitude of papers. Views are *sensitive to any change* in the underlying database. The notion of a snapshot, a type of materialized and *detached* view, was originally advanced by Adiba and Lindsay [2]. In contrast to snapshots, P-views are sensitive to insertions and logical deletions; and in contrast to views, P-views are insensitive to physical deletions.

Some work has studied various notions of derived data in an algebraic context (see, e.g., the compilations [6, 16]). Perhaps most closely related, Garcia-Molina et al. [5] explore how to "expire" (delete) data from materialized views so that a set of predefined, regular views on these materialized views are unaffected and can be maintained consistently with future updates. P-views solve a different problem and, e.g., do not involve two levels of views and do not assume a static set of predefined views. Finally, in the context of dimensional data warehouses, Skyt et al. [14] suggest a different approach for retaining aggregate data while eliminating detail data. This approach exploits the hierarchies in dimensions. The extended abstract [12] offers a brief overview of the paper's contributions.

The paper is structured as follows. The next section presents two example applications of P-views. As a concrete context for defining P-views, Section 3 presents the necessary parts of a temporal data model, a notion of physical deletion, vacuuming, and the formal definition of P-views. Section 4 presents a foundation for implementing P-views using so-called shadow relations. Finally, Section 5 summarizes and offers directions for future research. The detailed proof of correctness for the implementation strategy can be found in Appendix A, and a list of notation, used frequently throughout the paper, can be found in Appendix B.

2 Applications of P-Views

To illustrate the utility of P-views in append-only databases, a few possible applications are outlined.

The activity of analyzing click-stream data obtained from web servers is rapidly becoming an essential activity for e-businesses [7]: The purpose may be to do business analysis, user-behavior analysis [3], or to produce adaptive web sites [9, 10]. Because bulks of click-stream data are continuously being accumulated for such analyses, it is becoming increasingly important for IT departments to control the growth in data volumes.

Perhaps, the IT department wants to control the growth in data volumes by removing data more than 1 years old. This may be motivated by the observation that most of this data is inaccurate by now. However, the Marketing department wishes to trace business performance further back because they then can compare the effect of past marketing strategies during changes in the market.

A solution for the IT department is to use a disciplined physical deletion strategy, while simultaneously providing the Marketing department with a P-view with statistics on satisfaction measures, such as killed and completed sessions by time interval, or access patterns for each part of the web-site. This will enable the physical deletion of most data more than 1 year old, thus satisfying the need for growth control, while also satisfying the Marketing department.

Example: As a specific example, to be used throughout the paper, assume an e-business application that provides on-line news. Here, information is extracted from the click-stream data, is combined with customer and author information, and is then entered into the two relations *news* and *access* in Figure 1.

News items are represented, by *type of media* (i.e., whether it is plain text, XML, MPG video, audio, etc.,) *title*, *author name*, and *author address*. Similarly the user's *access* to news items is represented by

NewsId	*Media*	*Title*	*AuthorName*	*AuthorAddress*	$TT^{\vdash}$	$TT^{\dashv}$
1	'Video'	'Bosnia Today'	'M. Stone'	'California'	200	NOW
2	'XML'	'Bill Clinton'	'I. Cash'	'New York City'	300	NOW
3	'Text'	'Fall of EU'	'M. Stone'	'Washington DC'	305	349
4	'XML'	'Fall of EU'	'M. Stone'	'Washington DC'	350	NOW

AccessId	*NewsId*	*Domain*	*UserId*	VT	$TT^{\vdash}$	$TT^{\dashv}$
1	2	aol.com	16	303	303	NOW
2	2	whitehouse.gov	1201	303	303	NOW
3	1	aol.com	214	304	304	NOW
4	1	get2net.dk	512	304	304	NOW
5	1	eecs.cwru.edu	48	304	304	NOW
6	1	cs.auc.dk	198	305	305	NOW
7	3	dr-online.dk	3067	305	305	NOW
8	3	aol.com	347	305	305	NOW
9	1	eecs.cwru.edu	12	305	305	NOW
10	3	aol.com	201	305	305	NOW

Figure 1: Relations *news* and *access* at Time 660

newsid, *userid*, and *domain* (i.e., from which domain the access occurred.) It is recorded when an access took place (in VT); since the system cannot determine when an access stops, only one time is recorded. For both relations, attributes $TT^{\vdash}$ and $TT^{\dashv}$ record the transaction times of the items. For simplicity, the time unit is a day.

The IT department wants to physically delete access data registered more than one year ago. However, the Marketing department wants to trace the access patterns to different news articles. The following P-view will ensure that the necessary information is available.

$$\text{Define P-view P as: } \text{Agg}_{(\{NewsId, VT\}, NoOfAccess, \text{count}(access.AccessId))}(news \bowtie_{cond} access) \qquad (1)$$

where $cond$ denotes $news.NewsId = access.NewsId \wedge news.TT^{\vdash} \leq access.VT \leq news.TT^{\dashv}$. This query uses the aggregate formation operator, Agg [8]. It partitions the join result on attributes $\{NewsId, VT\}$, introduces a new attribute, $NoOfAccess$, and stores in this attribute in each tuple the result of computing count$(access.AccessId)$ on the group that the tuple participates in. Also, the aggregate formation operator projects on the grouping attributes and the new aggregate attribute, and it eliminates duplicates. The query thus computes for each news item and each day the number of accesses to the news item that day. P-views are formally defined in Section 3.3. ∎

Criminal records provide another application of P-views. The availability of such records is helpful in the investigation of unsolved crimes, and may also help composing effective crime prevention campaigns. For both purposes, all records are helpful independently of the specifics of the offense and the offender.

However, many rules influence what may be kept on record and how the information may be used. Specifically, records of certain minor offenses by juveniles must be removed when the offender reaches legal age. This conflicts with the general interest in crime prevention. For example, access to the records of minor offenses amongst juveniles will reveal which offenses are "popular," the areas with high concentration of offenses, and the backgrounds of the offenders.

These conflicting interests may be managed by applying P-views. Physical deletion can be specified based on the date-of-birth and the category of the offense. P-views can be defined that offer a variety of non-traceable statistics on minor offenses by year, age, background, as well as location and type of offense. The traceable data used for the aggregations will no longer be accessible.

3 Definition of Persistent Views

We proceed to precisely define P-views. First we introduce in turn the underlying relation structures and physical deletion.

3.1 Time and Temporal Relation Structures

Let T be a finite, non-empty set of times t with total order $<$. Also let NOW be a variable evaluating to the current time [4]. We use t_{now} for the time in T that corresponds to the current value of NOW, and we use T_{NOW} to denote the set T including the variable NOW. Next, we define the meaning of a time value in T_{NOW}.

DEFINITION 1 *For times $t \in T$ and $t' \in T_{NOW}$, the meaning, or value, of t' at time t, $[\![t']\!]_t$ is*

$$[\![t']\!]_t = \begin{cases} t & \text{if } t' = NOW \\ t' & \text{otherwise.} \end{cases}$$

Next, assume a set of non-empty domains and a set of attributes. Also, let $TT^{\vdash}$ and $TT^{\dashv}$ be distinguished attributes (capturing transaction time).

DEFINITION 2 *A temporal database schema is defined as a finite set of temporal relation schemas. A temporal relation schema is defined as a pair of (i) a set of user-defined attributes that together with the transaction-time attributes $TT^{\vdash}$ and $TT^{\dashv}$ are the attributes of the schema, and (ii) a function that assigns a domain to each attribute. Specifically it assigns domain T to attribute $TT^{\vdash}$, and domain T_{NOW} to attribute $TT^{\dashv}$.*

This definition follows those given in textbooks (e.g., [1]), with the exception that all relation schemas are temporal. The transaction-time attributes $TT^{\vdash}$ and $TT^{\dashv}$ are omni-present. We proceed to define database and relation instances.

DEFINITION 3 *A temporal database is a set of temporal relations; and a temporal relation is a finite set of tuples. A tuple u in a relation R is a function that for each attribute A, in the schema for R, assigns an element from the domain of A to attribute A. Specifically, it assigns an element in T to $TT^{\vdash}$, and an element in T_{NOW} to $TT^{\dashv}$, so that the following constraint is satisfied;*

$$\forall t' \geq u.TT^{\vdash}\ (u.TT^{\vdash} \leq [\![u.TT^{\dashv}]\!]_{t'} \wedge (u.TT^{\vdash} = t_{now} \Rightarrow u.TT^{\dashv} = NOW)).$$

A temporal relation is then a set of tuples, with each tuple being a function assigning values to the attributes such that the interval represented by $TT^{\vdash}$ and $TT^{\dashv}$ starts no later than it ends and such that $TT^{\dashv}$ is NOW if $TT^{\vdash}$ is the current time. A tuple u is *current at time t* if and only if $u.TT^{\vdash} \leq t \leq [\![u.TT^{\dashv}]\!]_t$ and simply *current* if it is current at t_{now}.

3.2 Physical Deletion

Transaction-time relations are effectively append only, with conventional deletions assuming a purely logical effect. We thus introduce a new facility for physical deletion, termed vacuuming. (Note that P-views are not dependent of the specific choice of physical deletion facility.)

Using vacuuming, specifications are stored in a special temporal relation that expresses what is to be physically deleted. Let *Vspec* be the attribute of this relation that ranges over a domain of vacuuming specification parts, v. This domain contains values of the form "$\omega(R)$: $\sigma_F(R)$," where R is a relation, σ is the standard selection operator, and F is a Boolean expression that may involve the attributes of R; symbol ω denotes either ρ or κ, specifying either a removal or a keep specification. The full syntax is given in Figure 2.

$$
\begin{array}{lll}
v & ::= & \omega(R) : Exp \\
\omega & ::= & \rho \mid \kappa \\
Exp & ::= & R \mid \sigma_F(Exp) \mid (Exp) \\
F & ::= & \texttt{true} \mid \texttt{false} \mid F\ bop\ F \mid \neg F \mid (F) \mid TT\ op\ tt \mid tt\ op\ TT \mid d\ op\ A_i \mid A_i\ op\ d \\
tt & ::= & t \mid s \mid tt - tt \mid tt + tt \mid (tt) \\
TT & ::= & TT^{\vdash} \mid TT^{\dashv} \\
bop & ::= & \vee \mid \wedge \\
op & ::= & < \mid > \mid = \mid \leq \mid \geq \mid \neq
\end{array}
$$

Figure 2: Syntax for Vacuuming Specification Parts (v)

A temporal vacuuming specification part, v, is a tuple assigning values from the domain described above to the *Vspec* attribute, and values from T and T_{NOW} to $TT^{\vdash}$ and $TT^{\dashv}$, respectively. A *temporal vacuuming specification* V is a temporal relation consisting of temporal vacuuming specification parts. Additional detail is offered elsewhere [11, 13].

Example: Continuing the example from Section 2, a removal specification part for relation $access$ is given next. In its lifetime, it specifies removal of the tuples that have a valid time (VT) with a value less or equal to 356 units less than current time, i.e., tuples time-stamped more than 356 time units (days) ago.

$$\rho(access) : \sigma_{VT \leq NOW - 356}(access)$$

■

The effect of a vacuuming specification V on a relation R is expressed in terms of the set of tuples remaining in R. This set is termed the *vacuumed relation*, denoted by (R, V).

DEFINITION 4 *Let $\{v_1, \ldots, v_k, v_{k+1}, \ldots, v_s\}$ be all the specification parts that concern R, i.e., parts with a Vspec value of the form "$\omega(R) : Exp$." Let $v_i \in \{v_1, \ldots, v_k\}$ be* removal *specification parts and $v_j \in \{v_{k+1}, \ldots, v_s\}$ be* keep *specification parts, where F_i and F_j are the time-dependent predicates of v_i and v_j. Then the vacuumed version of relation R at the current time, (R, V), is defined as*

$$(R, V) \stackrel{def}{=} \sigma_{\neg(\bigvee_{i=1}^{k} F_i) \vee (\bigvee_{j=k+1}^{s} F_j)}(R).$$

This states that a vacuumed relation is the original relation R where all data selected by any removal specification part ($\rho(R)$) is omitted, except from the data selected by a keep specification part ($\kappa(R)$). For a tuple to be selected by a vacuuming specification part, it must satisfy the selection criteria at some time during the lifetime of the specification part (see [11, 13] for further details).

DEFINITION 5 *A vacuumed temporal database is defined as a set of temporal relations vacuumed according to the vacuuming specification.*

3.3 P-Views

Sections 1 and 2 briefly stated and illustrated the goals of P-views. Consistent with those discussions, P-views do not interfere with physical deletion; but seen through a P-view, it should appear as if physical deletion has stopped at the time of definition of the P-view.

To make this precise, assume a database $db = \{R_1, R_2, \ldots, R_n\}$. Then this database at a time t is the set of component relations each in their respective states at this time, i.e., $[\![db]\!]_t = \{[\![R_1]\!]_t, [\![R_2]\!]_t, \ldots, [\![R_n]\!]_t\}$. $[\![R_i]\!]_t$ denotes the relation R_i as it was at time t, so it contains the tuples inserted into R_i before this time; and all tuples that were current at time t have the variable NOW as $TT^{\dashv}$-value. Let V denote the relation specifying the physical deletion.

Figure 3 gives the syntax of a P-view, $Pexp$. In the figure, L is a list of attribute names, C is a name

$Pexp$	::=	$\text{Agg}_{(L,C,func(A_i))}(Pexp) \mid Pexp \cup Pexp \mid Pexp \times Pexp \mid Pexp - Pexp \mid (Pexp) \mid$ $\sigma_F(Pexp) \mid \pi_L(Pexp) \mid R$
L	::=	$\{List\} \mid \emptyset$
$List$	::=	$A_i, List \mid A_i$
F	::=	$\texttt{true} \mid \texttt{false} \mid F\ bop\ F \mid \neg F \mid (F) \mid TT\ op\ tt \mid tt\ op\ TT \mid d\ op\ A_i \mid A_i\ op\ d$
tt	::=	$t \mid s \mid tt - tt \mid tt + tt \mid (tt)$
TT	::=	$TT^{\vdash} \mid TT^{\dashv}$
bop	::=	$\vee \mid \wedge$
op	::=	$< \mid > \mid = \mid \leq \mid \geq \mid \neq$

Figure 3: Syntax for P-views ($Pexp$).

given to the aggregate attribute achieved by using the aggregate function $func$ grouped on L, and A_i denotes an attribute name. Some conventional semantic constraints must also be followed. Let S be a finite, non-empty set of time spans, i.e., unanchored time intervals like 2 months. Then it is required that $d \in D_{A_i}$, $A_i \in U_A$, $t \in T_{NOW}$, and $s \in S$. For expressions such as $TT\ op\ tt$ and $tt\ op\ TT$, op should be defined for the domain T_{NOW} of $TT^{\vdash}$ and $TT^{\dashv}$, and tt should evaluate to an element in T_{NOW}. For expressions such as $A_i\ op\ d$ and $d\ op\ A_i$, op should be defined for the domain D_{A_i} of A_i, and $d \in D_{A_i}$. Finally, for $\sigma_F(Exp)$, F should only include attributes A_i in Exp.

Now, we can define the semantics of a P-view.

DEFINITION 6 *Let a P-view, given by "Define P-view P as $Pexp$," be entered into the database at time $Pexp.TT^{\vdash}$. Then the semantics of $Pexp$, at time $t \geq Pexp.TT^{\vdash}$ is defined as follows.*

$$[\![Pexp]\!]_t(db, V) \stackrel{def}{=} Pexp(([\![db]\!]_t, [\![V]\!]_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash} - 1]) \cup db_{overhead}), \tag{2}$$

where $db_{overhead}$ is any set of tuples satisfying the property

$$\forall u' \in db_{overhead}\ (u' \notin ([\![db]\!]_{Pexp.TT^{\vdash}}, [\![V]\!]_{Pexp.TT^{\vdash}}) \wedge \exists t'\ (t' < Pexp.TT^{\vdash} \wedge u' \in ([\![db]\!]_{t'}, [\![V]\!]_{t'}))). \tag{3}$$

This means that the P-view is defined as the expression $Pexp$ evaluated on the union of two elements.

The first of these denotes a database to which some vacuuming is applied. The database is db at time t, $[\![db]\!]_t$. The vacuuming applied to this database is given by $[\![V]\!]_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash}-1]$, which denotes V at time $Pexp.TT^{\vdash}-1$ where occurrences of NOW are replaced by $Pexp.TT^{\vdash}-1$. This stops all vacuuming as of this time.

Next, imagine that another P-view was defined prior to the definition of the P-view with expression $Pexp$. This earlier P-view will logically have stopped the vacuuming at the time of its definition. Rather than performing complex bookkeeping, a P-view is simply evaluated also on all such extra data. This is the data in $db_{overhead}$.

So a P-view is logically evaluated on the current database, where the applied vacuuming is terminated at the time the P-view is defined, union some data that derives from vacuuming having been terminated due to earlier P-views.

Example: The example from Section 2 involves two relations *news* and *access*, (see Figure 1 for the unvacuumed representations.) A vacuuming specification V is given next, and with this vacuuming in effect, the relation $access$ current at time 660 is as shown in Figure 4.

$$v = (\rho(access) : \sigma_{VT \leq NOW-356}(access), 400, NOW) \tag{4}$$

AccessId	*NewsId*	*Domain*	*UserId*	VT	$TT^{\vdash}$	$TT^{\dashv}$
6	1	cs.auc.dk	198	305	305	NOW
7	3	dr-online.dk	3067	305	305	NOW
8	3	aol.com	347	305	305	NOW
9	1	eecs.cwru.edu	12	305	305	NOW
10	3	aol.com	201	305	305	NOW

Figure 4: Relation $(access, V)$ at Time 660

Next, assume the P-view in Equation (1) in Section 2 is defined at time 500.

Had this been submitted as a regular query, the result would have been based on the unvacuumed relation *news* in Figure 1 and the vacuumed relation $(access, V)$ presented in Figure 4, resulting in the relation $\{(1, 305, 2), (3, 305, 3)\}$. However defined as a P-view, it is evaluated on the relations $(news, [\![V]\!]_{499}[NOW \leftarrow 499])$ and $(access, [\![V]\!]_{499}[NOW \leftarrow 499])$. Rewinding and terminating the vacuuming specification at time 499 gives the relation V containing the tuple shown below. Since the vacuuming specification part was defined before time 499, it is present, but has now been terminated.

$$v = (\rho(access) : \sigma_{VT \leq NOW-356}(access), 400, 499)$$

This means that only tuples with timestamps (VT) less than 143 ($499 - 356$) are vacuumed by the specification. This leaves the relation *access* as given in Figure 1, and the P-view will evaluate to the relation $\{(1, 304, 3), (1, 305, 2), (2, 303, 2), (3, 305, 3)\}$. ∎

4 Implementation Framework

Having defined P-views, we present a framework for implementing them. This framework uses so-called shadow relations, which are presented first. Then the predicates that control which tuples are entered into these shadow relations are derived. Finally, a correctness proof for the framework is described.

4.1 Retaining Data in Shadow Relations

The strategy for implementing vacuuming is to create a filter that hides the tuples that qualify for vacuuming, and then to actually physically delete these tuples in an asynchronous/lazy manner [11].

When adding support for P-views, instead of simply physically deleting a tuple, it needs to be evaluated if the tuple is necessary for evaluating a P-view; if so, it is necessary to retain the tuple. For this purpose, we equip each relation with a so-called shadow relation that has the same schema as that relation. Then P-views must be evaluated on the base relations as well as their shadow relations.

The decision procedure for physical deletion, $NewDelete$, is given next.

DEFINITION 7 *Let r be a set of tuples that are to be vacuumed from relation R, let R^S be the shadow relation of R, and let P^R be the predicate (to be specified later) that specifies all tuples that may be necessary for evaluating a P-view. Procedure $NewDelete$ is defined as follows.*

```
1)        PROCEDURE NewDelete (r, R) {
2)           IF (r ≠ ∅)
3)           THEN { select some u ∈ r
4)                  IF σ_{P^R}(R ∪ R^S) = σ_{P^R}((R ∪ R^S) − {u})
5)                  THEN { Delete (u, R) }
6)                  ELSE { Insert (u, R^S); Delete (u, R) }
7)                  NewDelete (r − {u}, R) } }
```

4.2 Specification of Shadow Relation Predicates

The overall framework for the specification is described, followed by base-step specifications and remaining specifications for selection and aggregate formation.

4.2.1 Overall Framework

The definition of $NewDelete$ uses the shadow relation predicate P^R on relation R, which specifies the tuples that may be necessary for evaluation of any P-view. This predicate is derived structurally from the P-view expressions that have been defined by the user.

DEFINITION 8 *We assume a state s containing a predicate P^R for each relation R in the database. Defining a P-view $Pexp$ affects the predicates P^R. We capture this effect in the* meaning *of an expression $Pexp$ as a partial function on states.*

$$S_{ds} : \mathit{Pexp} \rightarrow (\mathit{State} \hookrightarrow \mathit{State})$$

Initially, all P^R are false. *In Equations* (6)–(27)*, we define the semantics of a P-view expression $Pexp$ in terms of its structure (as specified in Figure 3). We let X, Y, and Z be P-view expressions and assume that all selection predicates are in Conjunctive Normal Form (CNF).*

The equations are presented in blocks, and some equations are explained through examples. As meta-syntax, we use $\star$ to denote *combination*. The application of each basic equation, Equations (6)–(8), essentially adds a disjunct to predicate P^R, rendering the order in which predicates are collected unimportant. Thus, *combination* is symmetric.

$$S_{ds}[\![X \star Y]\!]s = \begin{cases} s' & \text{if } \exists s_1, s_2\ ((S_{ds}[\![X]\!]s = s_1 \wedge S_{ds}[\![Y]\!]s_1 = s') \wedge \\ & \qquad (S_{ds}[\![Y]\!]s = s_2 \wedge S_{ds}[\![X]\!]s_2 = s')) \\ \mathit{undefined} & \text{otherwise} \end{cases} \tag{5}$$

We use $\star$ for combining the semantics of several P-view expressions, independently of the order of definition.

4.2.2 Base Steps for Selection and Aggregate Formation

We first define the base steps, which have a direct effect on the predicates collected in state s. Each base step introduces predicates stating, that tuples satisfying this predicate have to be saved in the shadow relation, to enable a correct evaluation of the P-views. Because tuples satisfying the predicates are indispensable, the predicates are combined with existing predicates using disjunction.

$$S_{ds}[\![R]\!]s = s[P^R \mapsto \text{true}] \tag{6}$$

$$S_{ds}[\![\sigma_p(R)]\!]s = s[P^R \mapsto (P^R) \vee (p)] \tag{7}$$

$$S_{ds}[\![\text{Agg}_{(L,C,\mathit{func}(A_i))}(R)]\!]s = \tag{8}$$

$$\begin{cases} s[P^R \mapsto (P^R) \vee (\mathit{min}(A_i))] & \text{if } L = \emptyset \wedge \mathit{func} = \text{“}\mathit{min}\text{”} \\ s[P^R \mapsto (P^R) \vee (\mathit{max}(A_i))] & \text{if } L = \emptyset \wedge \mathit{func} = \text{“}\mathit{max}\text{”} \\ s[P^R \mapsto (P^R) \vee (\mathit{min}(A_i \text{ by } B_1, \ldots, B_n))] & \text{if } L = \{B_1, \ldots, B_n\} \wedge \mathit{func} = \text{“}\mathit{min}\text{”} \\ s[P^R \mapsto (P^R) \vee (\mathit{max}(A_i \text{ by } B_1, \ldots, B_n))] & \text{if } L = \{B_1, \ldots, B_n\} \wedge \mathit{func} = \text{“}\mathit{max}\text{”} \\ s[P^R \mapsto \text{true}] & \text{otherwise} \end{cases}$$

As usual, $\sigma_{P^R}(R)$ is all tuples in R that satisfy P^R. When we apply P^R to a tuple, we replace all attribute names A_i in P^R with $u.A_i$. To make Equation (8) well defined, we define the values of the following min- and max-expressions for a tuple u in relation R.

$$\mathit{min}(u.A_i) = \begin{cases} \text{true} & \text{if } \forall u' \in R\ (u'.A_i \geq u.A_i) \\ \text{false} & \text{otherwise} \end{cases}$$

$$\mathit{max}(u.A_i) = \begin{cases} \text{true} & \text{if } \forall u' \in R\ (u'.A_i \leq u.A_i) \\ \text{false} & \text{otherwise} \end{cases}$$

$$\mathit{min}(u.A_i \text{ by } u.B_1, \ldots, u.B_n) = \begin{cases} \text{true} & \text{if } \forall u' \in R\ ((\forall j \in \{1, \ldots, n\}\ (u'.B_j = u.B_j)) \Rightarrow u'.A_i \geq u.A_i) \\ \text{false} & \text{otherwise} \end{cases}$$

$$\mathit{max}(u.A_i \text{ by } u.B_1, \ldots, u.B_n) = \begin{cases} \text{true} & \text{if } \forall u' \in R\ ((\forall j \in \{1, \ldots, n\}\ (u'.B_j = u.B_j)) \Rightarrow u'.A_i \leq u.A_i) \\ \text{false} & \text{otherwise} \end{cases}$$

Equations (6)–(8) provide the base cases for implementing P-views via shadow relations. We exemplify Equation (8).

Example: Building on the running example from Section 2, assume the following P-view is the only P-view defined on the relation access.

$$\text{Agg}_{(\{\mathit{NewsId}\}, \mathit{LastAccess}, \mathit{max}(VT))}(\mathit{access})$$

Before defining the P-view we have that $P^{access} = \text{false}$. Using Equation (8), the set of attributes L to group on is the single attribute $NewsId$ and is thus non-empty, and the function is "max". Thus, $s[P^{access} \mapsto \text{false} \vee max(VT \text{ by } NewsId)]$.

Assume the relation $access$ in Figure 1, an empty shadow relation $access^S$, and the vacuuming specification (4). Then at time 680, all the 10 tuples are chosen for vacuuming, and $NewDelete$ completely removes the tuples u satisfying the following (having omitted "false $\vee$ " from P^{access},) and moves the rest to the shadow relation.

$$\sigma_{max(VT \text{ by } NewsId)}(access \cup access^S) = \sigma_{max(VT \text{ by } NewsId)}((access \cup access^S) - \{u\})$$

Evaluating this on the tuples in the order of the $AccessId$-value results in the tuples $\{2, 9, 10\}$ being inserted into $access^S$. So, while relation $access$ is empty, its shadow relation is the one presented in Figure 5.

AccessId	*NewsId*	*Domain*	*UserId*	VT	$TT^{\vdash}$	$TT^{\dashv}$
2	2	whitehouse.gov	1201	303	303	NOW
9	1	eecs.cwru.edu	12	305	305	NOW
10	3	aol.com	201	305	305	NOW

Figure 5: Shadow Relation $access^S$ after Invocating $NewDelete$ at Time 680

■

4.2.3 Remaining Base Steps

The next part of the base semantics of P-views is expressed in Equations (9)–(13), covering the remaining operators $(\pi, (), \cup, \times, -)$. Each equation defines the semantics of one operator independently of the others.

$$S_{ds}[\![\pi_L(X)]\!]s = S_{ds}[\![X]\!]s \tag{9}$$
$$S_{ds}[\![(X)]\!]s = S_{ds}[\![X]\!]s \tag{10}$$
$$S_{ds}[\![Y \cup Z]\!]s = S_{ds}[\![Y \star Z]\!]s \tag{11}$$
$$S_{ds}[\![Y \times Z]\!]s = S_{ds}[\![Y \star Z]\!]s \tag{12}$$
$$S_{ds}[\![Y - Z]\!]s = S_{ds}[\![Y \star Z]\!]s \tag{13}$$

Equation (9) accounts for projections in a P-view expression. Since a shadow relation has the same schema as the corresponding base relation, and since the result of a projection may still include values from all tuples in the argument X, the equation states that the semantics of projection on X is the same as the semantics of X. Thus, projection does not eliminate any tuples.

Example: To illustrate, consider the P-view $\pi_{\{Domain\}}(access)$. Assuming that $P^{access} = \text{false}$, application of Equations (9) and (6) yield $S_{ds}[\![\pi_{\{Domain\}}(access)]\!]s = S_{ds}[\![access]\!]s = s[P^{access} \mapsto \text{true}]$. Thus, at time 680 where all 10 tuples in $access$ qualify for vacuuming, all 10 tuples will be inserted into $access^S$. ■

Equation (13) covers the use of set difference in P-views. To calculate a set difference, both the left and the right argument is needed. Thus, the semantics of $Y - Z$ is the semantics of Y combined with the semantics of Z.

Example: The following P-view illustrates Equation (13) and makes use of Equation (7) twice.

$$\sigma_{Domain=\text{'aol.com'}}(access) - \sigma_{NewsId=2}(access)$$

Using the equations, and assuming no other P-views, we optain the following derivation.

$$
\begin{aligned}
&S_{ds}[\![\sigma_{Domain=\text{'aol.com'}}(access) - \sigma_{NewsId=2}(access)]\!]s = \\
&S_{ds}[\![\sigma_{Domain=\text{'aol.com'}}(access) \star \sigma_{NewsId=2}(access)]\!]s = \\
&S_{ds}[\![\sigma_{NewsId=2}(access)]\!](s[P^{access} \mapsto \text{false} \vee Domain = \text{'aol.com'}]) = \\
&s[P^{access} \mapsto \text{false} \vee Domain = \text{'aol.com'} \vee NewsId = 2]
\end{aligned}
$$

Referring to Figure 1, using this predicate P^{access} in $NewDelete$ at time 680, where all 10 tuples are vacuumed, the tuples $\{1, 2, 3, 8, 10\}$ will form the shadow relation $access^S$. ∎

4.2.4 General Steps for Selection

The basic equations considered so far define operators σ and Agg based on a base relation R. Next, we define the semantics of selection in relation to a general expression $Pexp$. We cover the operator in relation to each of the 7 operators available. Specifically, Equations (14)-(20) define the semantics for $\sigma_p(Pexp)$.

$$S_{ds}[\![\sigma_{p_1}(\sigma_{p_2}(X))]\!]s = S_{ds}[\![\sigma_{p_1 \wedge p_2}(X)]\!]s \tag{14}$$
$$S_{ds}[\![\sigma_p(\text{Agg}_{(L,C,func(A_i))}(X))]\!]s = S_{ds}[\![\text{Agg}_{(L,C,func(A_i))}(X)]\!]s \tag{15}$$
$$S_{ds}[\![\sigma_p(\pi_L(X))]\!]s = S_{ds}[\![\sigma_p(X)]\!]s \tag{16}$$
$$S_{ds}[\![\sigma_p((X))]\!]s = S_{ds}[\![\sigma_p(X)]\!]s \tag{17}$$
$$S_{ds}[\![\sigma_p(Y \cup Z)]\!]s = S_{ds}[\![\sigma_p(Y) \star \sigma_p(Z)]\!]s \tag{18}$$
$$S_{ds}[\![\sigma_{p_{Y\times Z}}(Y \times Z)]\!]s = S_{ds}[\![\sigma_{p_Y}(Y) \star \sigma_{p_Z}(Z)]\!]s \tag{19}$$
$$S_{ds}[\![\sigma_p(Y - Z)]\!]s = S_{ds}[\![\sigma_p(Y) \star \sigma_p(Z)]\!]s \tag{20}$$

In Equation (19), $p_{Y\times Z}$ is required to be in CNF. Thus, $p_{Y\times Z} = p_Y \wedge p_Z \wedge p_{\{Y,Z\}}$, where p_Y, p_Z, and $p_{\{Y,Z\}}$ are in CNF. The conjuncts in p_Y only refer to attributes from expression Y, the conjuncts in p_Z only involve attributes from expression Z, and those in $p_{\{Y,Z\}}$ involve attributes from both Y and Z.

Equation (15) defines the semantics for a selection based on the result from the aggregate formation operator. This is illustrated next.

Example: Consider the P-view below.

$$\sigma_{(NewsId\geq 2)\wedge(LastAccess=303)}\text{Agg}_{(\{NewsId\},LastAccess,max(VT))}(access)$$

Its semantics is defined by Equations (15) and (8):

$$
\begin{aligned}
&S_{ds}[\![\sigma_{(NewsId\geq 2)\wedge(LastAccess=303)}\text{Agg}_{(\{NewsId\},LastAccess,max(VT))}(access)]\!]s = \\
&S_{ds}[\![\text{Agg}_{(\{NewsId\},LastAccess,max(VT))}(access)]\!]s = \\
&s[P^{access} \mapsto \text{false} \vee max(VT \text{ by } NewsId)]
\end{aligned}
$$

Using this predicate for P_{access} in $NewDelete$ results in tuples $\{2, 9, 10\}$ being retained in the shadow relation $access^S$. The P-view then correctly evaluates to $\{(2, 303)\}$.

Consider an alternative equation:

$$S_{ds}[\![\sigma_{p_{Agg}}(\text{Agg}_{(L,C,func(A_i))}(X))]\!]s = S_{ds}[\![\sigma_{p_X}(X) \star \text{Agg}_{(L,C,func(A_i))}(X)]\!]s \ ,$$

where p_X denotes the conjunct from p_{Agg} that refers only to attributes in X. Then the example evaluates to $s[P^{access} \mapsto \text{false} \vee NewsId \geq 2 \vee max(VT \text{ by } NewsId)]$, which would result in the tuples $\{1, 2, 7, 8, 9, 10\}$ being moved to the shadow relation. Thus the P-view would still evaluate to the correct result, but more tuples would be saved in the shadow relations. Thus the semantics in Equation (15) is preferable. ∎

Equation (19) defines the semantics for a selection based on a Cartesian product of two P-view expressions Y and Z. The selection predicate may involve expressions on attributes from Y, Z, or on a combination of their attributes. An example follows.

Example: The following P-view illustrates Equation (19).

$$\sigma_{(Domain=\text{'aol.com'}) \wedge (Media='XML') \wedge (access.VT \geq news.TT^{\vdash} \vee news.TT^{\dashv}=NOW) \wedge (access.NewsId=news.NewsId)}(access \times news)$$

Here, base relations $access$ and $news$ take the role of Y and Z, respectively. The predicate is in CNF and has three parts:

$$\begin{aligned} p_{access} &= \text{“}(Domain = \text{'aol.com'})\text{”} \\ p_{news} &= \text{“}(Media = 'XML')\text{”} \\ p_{\{access,news\}} &= \text{“}(access.VT \geq news.TT^{\vdash} \vee news.TT^{\dashv} = NOW) \wedge \\ &\qquad (access.NewsId = news.NewsId)\text{”} \end{aligned}$$

Each of these predicates must hold for a tuple from the Cartesian product to be indispensable. Therefore, when the predicates only on attributes in $access$, i.e., p_{access}, do not hold for the tuples $\{2, 4, 5, 6, 7, 9\}$, nor will they hold for those tuples in the Cartesian product that involve these. Thus, tuples $\{2, 4, 5, 6, 7, 9\}$ are disposable, and tuples $\{1, 3, 8, 10\}$ are possibly indispensable and are thus stored in $access^S$.

Similarly, if vacuuming is defined on base relation $news$, tuples $\{2, 4\}$ from that relation are possibly indispensable and are thus stored in $news^S$.

However, a predicate such as $(access.VT \geq news.TT^{\vdash} \vee news.TT^{\dashv} = NOW)$ cannot be evaluated based on tuples from one relation alone. The first part clearly involves tuples from both relations. Thus, not all predicates can be used directly on Y or Z to separate the disposable tuples from the indispensable ones.

The semantics of the P-view is derived as follows using Equations (19) and (7).

$$\begin{aligned} &S_{ds}[\![\sigma_{(Domain=\text{'aol.com'}) \wedge (Media='XML') \wedge (access.VT \geq news.TT^{\vdash} \vee news.TT^{\dashv}=NOW) \wedge (access.NewsId=news.NewsId)}(access \times news)]\!]s = \\ &S_{ds}[\![\sigma_{Domain=\text{'aol.com'}}(access) \star \sigma_{Media='XML'}(news)]\!]s = \\ &S_{ds}[\![\sigma_{Domain=\text{'aol.com'}}(access)]\!](s[P^{news} \mapsto \text{false} \vee Media = 'XML']) = \\ &s[P^{news} \mapsto \text{false} \vee Media = 'XML', P^{access} \mapsto \text{false} \vee Domain = \text{'aol.com'}] \end{aligned}$$

■

4.2.5 General Steps for Aggregate Formation

The last group of equations concern the aggregate formation operator. These are defined in Equations (21)–(27).

$$S_{ds}[\![\text{Agg}_{(L,C,func(A_i))}(\sigma_p(X))]\!]s = S_{ds}[\![\sigma_p(X)]\!]s \tag{21}$$

$$S_{ds}[\![\text{Agg}_{(L_1,C_1,func_1(A_i))}(\text{Agg}_{(L_2,C_2,func_2(A_j))}(X))]\!]s = S_{ds}[\![\text{Agg}_{(L_2,C_2,func_2(A_j))}(X)]\!]s \tag{22}$$

$$S_{ds}[\![\text{Agg}_{(L_1,C_1,func_1(A_i))}(\pi_{L_2}(X))]\!]s = S_{ds}[\![\text{Agg}_{(L_1,C_1,func_1(A_i))}(X)]\!]s \tag{23}$$

$$S_{ds}[\![\text{Agg}_{(L,C,func(A_i))}((X))]\!]s = S_{ds}[\![\text{Agg}_{(L,C,func(A_i))}(X)]\!]s \tag{24}$$

$$S_{ds}[\![\text{Agg}_{(L,C,func(A_i))}(Y \cup Z)]\!]s = S_{ds}[\![\text{Agg}_{(L,C,func(A_i))}(Y) \star \text{Agg}_{(L,C,func(A_i))}(Z)]\!]s \tag{25}$$

$$S_{ds}[\![\text{Agg}_{(L,C,func(A_i))}(Y - Z)]\!]s = S_{ds}[\![Y - Z]\!]s \tag{26}$$

$$S_{ds}[\![\text{Agg}_{(L,C,func(A_i))}(Y \times Z)]\!]s =$$
$$\begin{cases} S_{ds}[\![\text{Agg}_{(L,C,func(A_i))}(Y) \star Z]\!]s & \text{if } A_i \in A_Y \ \wedge \ L \cap A_Z = \emptyset \\ S_{ds}[\![Y \star \text{Agg}_{(L,C,func(A_i))}(Z)]\!]s & \text{if } A_i \in A_Z \ \wedge \ L \cap A_Y = \emptyset \\ S_{ds}[\![Y \times Z]\!]s & \text{otherwise} \end{cases} \quad (27)$$

In Equation (27), A_Y is the set of attributes in Y and A_Z is the set of attributes in Z. Note that in Equation (21), using also the semantics from the aggregation would at most result in maintaining more tuples in the shadow relations than with the current definition, only tuples in the selection result are used in the aggregation. A similar line of reasoning underlies the design of Equations (22) and (26).

Equation (26) defines the semantics for the aggregate formation operator based on the set difference between two expressions, Y and Z. The following example explains this equation.

Example: Consider the following P-view and the relations $access_1$ and $access_2$ shown in Figures 6 and 7.

$$\text{Agg}_{(\{NewsId\},FirstAccess,min(VT))}(access_1 - access_2)$$

AccessId	*NewsId*	*Domain*	*UserId*	VT	$TT^{\vdash}$	$TT^{\dashv}$
3	1	aol.com	214	304	304	NOW
4	1	get2net.dk	512	304	304	NOW
5	1	eecs.cwru.edu	48	304	304	NOW
6	1	cs.auc.dk	198	305	305	NOW
7	3	dr-online.dk	3067	305	305	NOW
8	3	aol.com	347	305	305	NOW
9	1	eecs.cwru.edu	12	305	305	NOW
10	3	aol.com	201	305	305	NOW

Figure 6: Relation $access_1$

AccessId	*NewsId*	*Domain*	*UserId*	VT	$TT^{\vdash}$	$TT^{\dashv}$
1	2	aol.com	16	303	303	NOW
2	2	whitehouse.gov	1201	303	303	NOW
3	1	aol.com	214	304	304	NOW
4	1	get2net.dk	512	304	304	NOW
5	1	eecs.cwru.edu	48	304	304	NOW

Figure 7: Relation $access_2$

Using Equations (26), (6), and (13) we obtain the predicates P^{access_1} and P^{access_2} as described next.

$$S_{ds}[\![\text{Agg}_{(\{NewsId\},FirstAccess,min(VT))}(access_1 - access_2)]\!]s =$$
$$S_{ds}[\![access_1 - access_2]\!]s =$$
$$S_{ds}[\![access_1 \star access_2]\!]s =$$
$$s[P^{access_1} \mapsto \text{true}, P^{access_2} \mapsto \text{true}]$$

Thus, all tuples in the two relations will be moved to the shadow relations, and evaluation of the aggregate formation operator will be based on the correct set of tuples, $\{6, 7, 8, 9, 10\}$.

Let us consider the effect of (wrongly!) letting the semantics allow us to take the aggregation into account on Y, Z, or, both of these. For Y, we would obtain the following equation.

$$S_{ds}[\![\text{Agg}_{(L,C,func(A_i))}(Y - Z)]\!]s = S_{ds}[\![\text{Agg}_{(L,C,func(A_i))}(Y) \star Z]\!]s$$

The correct result of evaluating the P-view is $\{(1, 305), (3, 305)\}$.

Y: Allowing the basic semantics on Y, i.e., $access_1$, we would get

$$s[P^{access_1} \mapsto min(VT \text{ by } NewsId), P^{access_2} \mapsto \text{true}]$$

This will result in the shadow relations $access_1^S = \{5, 10\}$ and $access_2^S = \{1, 2, 3, 4, 5\}$. The P-view would then evaluate to $\{(3, 305)\}$, since the set difference will return only tuple 10.

Z: Allowing the basic semantics on Z, i.e., $access_2$, we would get

$$s[P^{access_1} \mapsto \text{true}, P^{access_2} \mapsto min(VT \text{ by } NewsId)]$$

This will result in the shadow relations $access_1^S = \{3, 4, 5, 6, 7, 8, 9, 10\}$ and $access_2^S = \{2, 5\}$. The P-view evaluates to $\{(1, 304), (3, 305)\}$, based on the tuples $\{4, 10\}$.

Y/Z: Allowing the basic semantics on both $access_1$ and $access_2$, we would get

$$s[P^{access_1} \mapsto min(VT \text{ by } NewsId), P^{access_2} \mapsto min(VT \text{ by } NewsId)]$$

This will result in the shadow relations $access_1^S = \{5, 10\}$ and $access_2^S = \{2, 5\}$. The P-view now evaluates to $\{(3, 305)\}$, based on tuple 10.

Thus, the aggregate formation operator gives no useful knowledge regarding whether tuples from either $access_1$ or $access_2$ are disposable or indispensable. ∎

4.3 Correctness

The theorem that follows states that the proposed mechanism for accumulating data in shadow relations enables the system to correctly compute P-views.

THEOREM 1 *Assume that the NewDelete-procedure is used for removing vacuumed data, as described throughout Sections 4.1 and 4.2.*

Then for all vacuuming specifications V, databases db, P-views $Pexp$, and times $t \geq Pexp.TT^{\vdash}$, the following holds.

$$Pexp([\![db]\!]_t \cup [\![db^S]\!]_t) = [\![Pexp]\!]_t(db, V) \tag{28}$$

PROOF Applying the definition of P-views (Definition 6) to the right-hand side of Equation (28), we see that to prove Theorem 1 we need to *show the existence of an overhead $db_{overhead}$* such that Equation (28) is satisfied. We choose $db_{overhead} = \{R_1^O, \ldots, R_n^O\}$, where each R_i^O has the same schema as a relation R_i in *db*, and where

$$\begin{aligned} R_i^O = \{u \mid \exists t' < Pexp.TT^{\vdash}\ (u \in ([\![R_i]\!]_{t'-1}, [\![V]\!]_{t'-1}) \wedge u \notin ([\![R_i]\!]_{t'}, [\![V]\!]_{t'}) \wedge \\ \sigma_{[\![P^{R_i}]\!]_{t'}}(R_i \cup R_i^S) \neq \sigma_{[\![P^{R_i}]\!]_{t'}}((R_i \cup R_i^S) - \{u\}))\}. \end{aligned} \tag{29}$$

Here, $(\llbracket R_i \rrbracket_{t'}, \llbracket V \rrbracket_{t'})$ and $(\llbracket R_i \rrbracket_{t'-1}, \llbracket V \rrbracket_{t'-1})$ denote the relation R_i vacuumed at time t' and $t'-1$, respectively. The predicate $\llbracket P^{R_i} \rrbracket_{t'}$ is the shadow relation predicate for relation R_i constructed from the P-views at time t' as specified in Definition 8.

To see that this is a valid choice, note that for any tuple $u \in db_{overhead}$ there exists a corresponding relation R_i from where u was removed at time t', i.e., before the time $Pexp.TT^{\vdash}$. Therefore, u is not there at time $Pexp.TT^{\vdash}$, but was there at an earlier time $t'-1$. Thus, $db_{overhead}$ is a set of tuples satisfying the property stated in Definition 6. Furthermore, as chosen, a tuple u in $db_{overhead}$ was vacuumed at time t', and because $\sigma_{\llbracket P^{R_i} \rrbracket_{t'}}(R_i \cup R_i^S) \neq \sigma_{\llbracket P^{R_i} \rrbracket_{t'}}((R_i \cup R_i^S) - \{u\})$, i.e., the criteria used by $NewDelete$, u was saved in the shadow relation. Therefore, $db_{overhead}$ is the exact set of tuples saved in db^S before time $Pexp.TT^{\vdash}$ due to the existence of P-views.

Having chosen $db_{overhead}$ like this, we show Theorem 1 in two steps.

STEP 1) For the argument databases on the left- and right-hand side of Equation 28 (having substituted the right-hand side by the definition of P-views, Definition 6,) show the inclusion

$$(\llbracket db \rrbracket_t \cup \llbracket db^S \rrbracket_t) \subseteq ((\llbracket db \rrbracket_t, \llbracket V \rrbracket_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash} - 1]) \cup db_{overhead}). \tag{30}$$

This shows that the physical database achieved by using the shadow relations is contained in the database used in Definition 6.

STEP 2) If we denote the tuples that appear on the right-hand side but not on the left-hand side of inclusion (30) as *excess tuples*, then show that no excess tuple u will influence the evaluation of the P-view, i.e., that

$$\begin{array}{l} u \in ((\llbracket db \rrbracket_t, \llbracket V \rrbracket_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash} - 1]) \cup db_{overhead}) \wedge \\ \quad u \notin (\llbracket db \rrbracket_t \cup \llbracket db^S \rrbracket_t) \\ \Downarrow \\ Pexp((\llbracket db \rrbracket_t, \llbracket V \rrbracket_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash} - 1]) \cup db_{overhead}) = \\ Pexp(((\llbracket db \rrbracket_t, \llbracket V \rrbracket_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash} - 1]) \cup db_{overhead}) - \{u\}). \end{array} \tag{31}$$

First, all tuples in $db_{overhead}$ also belong to $\llbracket db^S \rrbracket_t$, so no tuple in $db_{overhead}$ is an excess tuple.

Using this, we show that all excess tuples have the property

$$\begin{array}{l} \exists t' \geq Pexp.TT^{\vdash}\ (\neg Pvac_{t'-1}(u) \wedge Pvac_{t'}(u) \wedge \\ \qquad\qquad \sigma_{\llbracket P^R \rrbracket_{t'}}(R \cup R^S) = \sigma_{\llbracket P^R \rrbracket_{t'}}((R \cup R^S) - \{u\})). \end{array} \tag{32}$$

This property states, that an excess tuple is vacuumed at a time t' after time $Pexp.TT^{\vdash}$ and that the $NewDelete$-procedure, based on the shadow relation predicate P^R which was collected for relation R as explained throughout Section 4.2, did not save it in a shadow relation.

Finally, using induction in the structure of the predicate P^R, we show implication (31), i.e., that no excess tuple influence the evaluation of the P-view, $Pexp$.

The two steps prove that choosing $db_{overhead}$ as above in Equation (29), a tuple removed from the database by the $NewDelete$ procedure (within the lifetime of a P-view) has no influence on the evaluation of the P-view. This shows that the implementation strategy is correct. Q.E.D.

For the interested reader the detailed proof, containing the proofs of STEP 1) and STEP 2), is presented in Appendix A.

4.4 Relationship to Views and Snapshots

Having defined P-views and an accompanying implementation framework, we proceed to illustrate how P-views differ from traditional views and snapshots, and we emphasize the independence of P-views on the specific mechanism chosen for physical deletion. We use the example from Section 2 to explore how the need for physical deletion and the need for summary data may be accommodated simultaneously using existing mechanisms. The available mechanisms are vacuuming in conjunction with either traditional views or snapshots [2].

First, assume a traditional view is used in place of the P-view. Then the physical deletion mechanism (i.e., vacuuming) needs to be adjusted to enable computing the view correctly; otherwise, the view is affected and information lost when vacuuming occurs (see the example below). To avoid such a loss, new keep specifications may be entered or removal specifications may be changed. However, this is not an attractive solution, since the adjustments tend to be very difficult, involving complicated specifications. Alternatively, the vacuuming will end up very "imprecise," leading to little actual physical removal. Also, the potential for physical removal is compromised, since a correct computation of a traditional view is based on the base relations. Finally, the legal requirements for removing data are not met—the extra data will be retained in the base relations and will be accessible at least to the database administrator. In conclusion, combining vacuuming and traditional views is inadequate.

Example: Assume the P-view with schema $\{NewsId, VT, NoOfAccess\}$, specified in Equation (1) is created at time 350, and evaluates to $\{(1, 304, 3), (1, 305, 2), (2, 303, 2), (3, 305, 3)\}$. Assume that the vacuuming specification in Equation (4) takes effect at time 400. At this moment, only tuples having their VT value less than or equal to 44 $(400 - 356)$ are removed, so no tuples are removed, and the view is intact. But at time 660, the tuples with VT value less than or equal to 304 are absent. Five tuples are affected (see Figure 1), and when the view is recomputed, only the five tuples remaining in *access* will be considered, yielding the resulting relation $\{(1, 305, 2), (3, 305, 3)\}$. We have thus lost the desired access to summary data. ∎

As another alternative, assume that a snapshot [2] is used in place of the P-view. The snapshot comprises a static picture of the answer to the query expression at the time of its creation. Even after introducing vacuuming, the snapshot will as desired remain unchanged. The problem is that when a new tuple is inserted into the relation, this tuple will not be reflected in the result. Also, creating the snapshot again will not produce the correct result since data has been vacuumed.

In conclusion, traditional views and snapshots fall short in meeting the application's needs. The proposed P-views aim to meet these unmet needs.

Finally, it is important to observe that P-views are independent of the mechanism chosen for physical deletion. All the framework requires is to be able to "see" a tuple before it is physically deleted. This enables the $NewDelete$ algorithm (Definition 7) to apply its decision procedure and possibly place the deleted tuple in the appropriate shadow relation, which then ensures that the P-views are computed correctly.

5 Summary and Research Directions

Motivated by the need for flexible mechanisms to manage the growing amounts of aged or obsolete data and based on the observation that a wide range of applications—e.g., financial and medical application and applications in e-business and data warehousing—rely on append-only databases, the paper introduces a new kind of view, termed *persistent views*, or P-views for short.

In append-only databases, deletion has a logical effect only; all past database states are retained, with the result that data volumes grow monotonically. New physical deletion facilities, termed vacuuming, are

introduced. P-views are similar to conventional views, with the exception that physical deletions have no effect on P-views. Although definition-wise the difference between regular views and P-views is small, the implications of this difference are profound. We emphasize the following.

- P-views allow to delete and weed out the detail data while retaining select and aggregate information.
- P-view enable access to anonymous aggregate informations, when deletion of the detail data is required.
- P-views offer access control on detail data.

The paper shows how P-views are quite useful for eliminating bulks of detailed, old, and inaccurate data, while preserving only select or aggregate data. Specifically, one may specify P-views that retrieve the desired, e.g., aggregate, data from the base relations and then physically delete all detailed data from the base relations. In addition, P-views is a general mechanism that has applications beyond the focus of this paper.

When physically deleting base data, it is generally necessary to retain some of this data transparently to the user in order to be able to compute the P-views. The paper proposes a mechanism for accomplishing this retention using so-called shadow relations, thus offering a systematic approach to implementing P-views.

In future research, it would be of interest to further refine the foundation for implementing P-views, since the current foundation retains more data in its shadow relations than is strictly necessary for computing the P-views. Most prominently, projections in P-views are not exploited to retain less data. This may possibly be achieved by introducing multiple shadow relations per base relation; a more radical change would be to abolish the shadow relations altogether and instead use relations that are tied to the individual P-views or subexpressions of P-views. In addition, it would be of interest to prototype the foundation and investigate performance issues relevant to the implementation strategy.

References

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

[2] M. E. Adiba and B. G. Lindsay. Database Snapshots. In *Proceedings of the 6th International Conference on Very Large Databases*, pages 86–91, 1980.

[3] J. Andersen, A. Giversen, A. H. Jensen, R. S. Larsen, T. B. Pedersen, and J. Skyt. Analyzing Clickstreams Using Subsessions. In *Proceedings of the 3rd International Workshop on Data Warehousing and OLAP*, pages 25–32, 2000

[4] J. Clifford, C. Dyreson, T. Isakowitz, C. S. Jensen, and R. T. Snodgrass. On the Semantics of "Now" in Databases. *ACM Transactions on Database Systems*, 22(2):171–214, 1997.

[5] H. Garcia-Molina, W. Labio, and J. Yang. Expiring Data in a Warehouse. In *Proceedings of the 24th International Conference on Very Large Databases*, pages 500–511, 1998.

[6] A. Gupta and I. S. Mumick (editors). *Materialized Views—Techniques, Implementations, and Applications.* MIT Press, 1999.

[7] R. Kimball. *The Data Webhouse Toolkit*. Wiley, 2000.

[8] A. Klug. Equivalence of Relational Algebra and Relational Calculus Query Languages Having Aggregate Functions. *Journal of the ACM*, 29(3):699–717, 1982.

[9] M. Perkowitz and O. Etzioni. Adaptive Sites: Automatically Learning From User Access Patterns. In *Proceedings of the 6th International World Wide Web Conference*, poster no. 722, 1997.

[10] M. Perkowitz and O. Etzioni. Towards Adaptive Web Sites: Conceptual Framework and Case Study. In *Proceedings of the 8th International World Wide Web Conference/Computer Networks*, 31(11-16):1245–1258, 1999.

[11] J. Skyt and C. S. Jensen. Vacuuming Temporal Databases. TIMECENTER Technical Report, TR-32, `www.cs.auc.dk/TimeCenter/pub.htm`, 20 pages, September 1998.

[12] J. Skyt and C. S. Jensen. Managing Aging Data Using Persistent Views (extended abstract). In *Proceedings of the 7th International Conference on Cooperative Information Systems*, pages 132–137, 2000. Short version of this paper.

[13] J. Skyt, C. S. Jensen, and L. Mark. A Foundation for Vacuuming Temporal Databases. Submitted to Data and Knowledge Engineering, 35 pages, October 2000. Extended version of [11].

[14] J. Skyt, C. S. Jensen, and T. B. Pedersen. Specification-Based Data Reduction in Dimensional Data Warehouses. TIMECENTER Technical Report, TR-61, `www.cs.auc.dk/TimeCenter/pub.htm`, 31 pages, July 2001.

[15] R. T. Snodgrass and I. Ahn. Temporal databases. *IEEE Computer*, 19(9):35–42, 1986.

[16] J. Widom (editor). Special Issue on Materialized Views and Data Warehousing. *IEEE Data Engineering Bulletin*, 18(2), 1995.

A Proof of Theorem 1

This appendix presents a detailed proof of Theorem 1 according to the outline presented in Section 4.3. To set the stage, first we repeat the theorem.

THEOREM 1 (REPEATED) *Assume that the $NewDelete$-procedure is used for removing vacuumed data, as described throughout Sections 4.1 and 4.2.*

Then for all vacuuming specifications V, databases db, P-views $Pexp$, and times $t \geq Pexp.TT^{\vdash}$, the following holds.

$$Pexp(\llbracket db \rrbracket_t \cup \llbracket db^S \rrbracket_t) = \llbracket Pexp \rrbracket_t(db, V) \tag{33}$$

Before outlining the proof, we recall some notation.

- Let t_{rem} denote the time of the most recent invocation of procedure $NewDelete$.
- A relation R vacuumed at time t is defined as $(\llbracket R \rrbracket_t, \llbracket V \rrbracket_t) = \sigma_{\neg(\bigvee_{i=1}^{k} F_i) \vee (\bigvee_{j=k+1}^{s} F_j)}(R)$, where F_i and F_j are the time-dependent predicates of removal and keep specification parts, respectively [11, 13]. Thus, a tuple satisfying the predicate $(\bigvee_{i=1}^{k} F_i \wedge \neg \bigvee_{j=k+1}^{s} F_j)$ is selected for vacuuming at or before time t. We denote this predicate $Pvac_t$.
- With $Pexp.TT^{\vdash}$ as the time of definition of a P-view $Pexp$, let P_V be the set of P-views $PV_1, \ldots, PV_r$ that are all current in db at time $Pexp.TT^{\vdash}$.

  Recall that a shadow relation predicate, P^R for a relation R, is collected into a state s as described throughout Section 4.2, and assume that s' is the state of false predicates, i.e., $P^R = \text{false}$, for all relations R. Then according to Definition 8 the state current at time $Pexp.TT^{\vdash}$ is a set of predicates P^R as specified in the following.

$$S_{ds}[\![PV_1 \star \cdots \star PV_r \star Pexp]\!]s' = \{[\![P^R]\!]_{Pexp.TT^{\vdash}} | R \in \llbracket db \rrbracket_{Pexp.TT^{\vdash}}\} \tag{34}$$

- When the $NewDelete$ procedure is invoked at time t, it removes a tuple u from relation R *without* storing a copy in the shadow relation R^S *if and only if* $\sigma_{[\![P^R]\!]_t}(R \cup R^S) = \sigma_{[\![P^R]\!]_t}((R \cup R^S) - \{u\})$.

PROOF Due to the definition of P-views (Definition 6) proving Theorem 1 equals proving Equations (35) and (36). Here the right-hand side of Equation (33) is replaced by the definition of the P-view.

$$\begin{aligned} &Pexp(\llbracket db \rrbracket_t \cup \llbracket db^S \rrbracket_t) = \\ &Pexp((\llbracket db \rrbracket_t, \llbracket V \rrbracket_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash} - 1]) \cup db_{overhead}), \end{aligned} \tag{35}$$

where $db_{overhead}$ is any set of tuples satisfying the property

$$\begin{aligned} \forall u' \in db_{overhead}\ (u' \notin (\llbracket db \rrbracket_{Pexp.TT^{\vdash}}, \llbracket V \rrbracket_{Pexp.TT^{\vdash}}) \wedge \\ \exists t'\ (t' < Pexp.TT^{\vdash}\ \wedge\ u' \in (\llbracket db \rrbracket_{t'}, \llbracket V \rrbracket_{t'}))). \end{aligned} \tag{36}$$

Now, *if we show the existence of an overhead $db_{overhead}$* such that Equation (35) and (36) are satisfied, we have proved Theorem 1.

OUTLINE

We let $db_{overhead} = \{R_1^O, \ldots, R_n^O\}$, where each R_i^O has the same schema as relation R_i in db, and where

$$R_i^O = \{u \mid \exists t' < Pexp.TT^{\vdash} \ (u \in (\llbracket R_i \rrbracket_{t'-1}, \llbracket V \rrbracket_{t'-1}) \wedge u \notin (\llbracket R_i \rrbracket_{t'}, \llbracket V \rrbracket_{t'}) \wedge \\ \sigma_{\llbracket P^{R_i} \rrbracket_{t'}}(R_i \cup R_i^S) \neq \sigma_{\llbracket P^{R_i} \rrbracket_{t'}}((R_i \cup R_i^S) - \{u\}))\}. \tag{37}$$

Here, $(\llbracket R_i \rrbracket_{t'}, \llbracket V \rrbracket_{t'})$ and $(\llbracket R_i \rrbracket_{t'-1}, \llbracket V \rrbracket_{t'-1})$ denote the relation R_i vacuumed at time t' and $t'-1$, respectively. The predicate $\llbracket P^{R_i} \rrbracket_{t'}$ is the shadow relation predicate for relation R_i constructed from the P-views at time t' as specified in Definition 8.

To see that this is a valid choice, note that for any tuple $u \in db_{overhead}$ there exists a corresponding relation R_i from where u was removed at time t', i.e., before time $Pexp.TT^{\vdash}$. Therefore, u is not there at the time $Pexp.TT^{\vdash}$, but was there at an earlier time $t'-1$. Thus, $db_{overhead}$ is a set of tuples satisfying the property stated in Definition 6 and Equation (36). Furthermore, as chosen, u was removed from R_i at time t' due to vacuuming, and because $\sigma_{\llbracket P^{R_i} \rrbracket_{t'}}(R_i \cup R_i^S) \neq \sigma_{\llbracket P^{R_i} \rrbracket_{t'}}((R_i \cup R_i^S) - \{u\})$, i.e. the criteria used by $NewDelete$, u was saved in the shadow relation. Therefore, $db_{overhead}$ is the exact set of data saved in db^S before time $Pexp.TT^{\vdash}$ due to the existence of P-views.

Haven chosen $db_{overhead}$ like this, we show Equation (35) in two steps.

STEP 1. For the argument databases on the left- and right-hand side of Equation 35, show the following inclusion.

$$(\llbracket db \rrbracket_t \cup \llbracket db^S \rrbracket_t) \subseteq ((\llbracket db \rrbracket_t, \llbracket V \rrbracket_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash} - 1]) \cup db_{overhead}) \tag{38}$$

This shows that the physical database achieved by using the shadow relations is contained in the database used in Definition 6.

STEP 2. For the *excess tuples* u, i.e., the tuples that appear on the right-hand side but not on the left-hand side of inclusion (38), show that no such excess tuple u will influence the evaluation of the P-view, i.e., show that

$$\begin{array}{l} u \in ((\llbracket db \rrbracket_t, \llbracket V \rrbracket_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash} - 1]) \cup db_{overhead}) \wedge \\ \quad u \notin (\llbracket db \rrbracket_t \cup \llbracket db^S \rrbracket_t) \\ \Downarrow \\ Pexp((\llbracket db \rrbracket_t, \llbracket V \rrbracket_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash} - 1]) \cup db_{overhead}) = \\ Pexp(((\llbracket db \rrbracket_t, \llbracket V \rrbracket_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash} - 1]) \cup db_{overhead}) - \{u\}). \end{array} \tag{39}$$

The two steps prove that choosing $db_{overhead}$ as above in Equation (37), a tuple removed from the database by the $NewDelete$ procedure (within the lifetime of a P-view) has no influence on the evaluation of the P-view. This shows that the implementation strategy is correct.

STEP 1)

We prove STEP 1 by proving each of the following.

$$u \in \llbracket db \rrbracket_t \Rightarrow (u \in (\llbracket db \rrbracket_t, \llbracket V \rrbracket_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash} - 1]) \vee u \in db_{overhead}) \tag{40}$$

$$u \in \llbracket db^S \rrbracket_t \Rightarrow (u \in (\llbracket db \rrbracket_t, \llbracket V \rrbracket_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash} - 1]) \vee u \in db_{overhead}) \tag{41}$$

First we show (40). Since $u \in [\![db]\!]_t$, no vacuuming has yet caused its removal from relation R_i at time t. This is the case in two situations: a) u has not been vacuumed at time t, i.e., $\neg Pvac_t(u)$, and b) u satisfies the vacuuming predicate at some time t' after the latest invocation of $NewDelete$, i.e., $\exists t' \ (t_{rem} < t' \leq t \wedge Pvac_{t'}(u))$. In both cases, $u \in ([\![db]\!]_t, [\![V]\!]_{t_{rem}}[NOW \leftarrow t_{rem}])$. Thus, since $Pexp.TT^{\vdash} \leq t_{rem}$ we conclude that

$$u \in ([\![db]\!]_t, [\![V]\!]_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash} - 1]).$$

Then we show (41). Since $u \in [\![db^S]\!]_t$, u was vacuumed at some time t' before the latest invocation of $NewDelete$, and u showed at that time to be indispensable due to a P-view, i.e., $\exists t' \leq t_{rem} \ (Pvac_{t'}(u) \wedge \sigma_{[\![P^{R_i}]\!]_{t'}}(R_i \cup R_i^S) \neq \sigma_{[\![P^{R_i}]\!]_{t'}}((R_i \cup R_i^S) - \{u\}))$.

If $t' \geq Pexp.TT^{\vdash}$, it is given that u was vacuumed after $Pexp$'s definition. Thus, we know that $u \in ([\![db]\!]_t, [\![V]\!]_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash} - 1])$, since this is exactly the database where vacuuming is stopped completely at the time of $Pexp$'s definition.

Alternatively, $t' < Pexp.TT^{\vdash}$. We know that $u \in ([\![R_i]\!]_{t'-1}, [\![V]\!]_{t'-1}) \wedge u \notin ([\![R_i]\!]_{t'}, [\![V]\!]_{t'})$ since t' was the time of vacuuming. Also, since u was inserted into R_i^S, there must have been a predicate P^{R_i} such that $\sigma_{[\![P^{R_i}]\!]_{t'}}(R_i \cup R_i^S) \neq \sigma_{[\![P^{R_i}]\!]_{t'}}((R_i \cup R_i^S) - \{u\})$. Therefore $u \in R_i^O$ and $u \in db_{overhead}$.

In conclusion, STEP 1 follows.

STEP 2)

When proving that all *excess tuples* on the right-hand side of (38) are disposable for the evaluation of the P-view, we first show the following.

$$u \in db_{overhead} \Rightarrow u \in [\![db^S]\!]_t \tag{42}$$

This shows that no tuple in $db_{overhead}$ is an excess tuple on the right-hand side of (38). Thus, it narrows the specification of the excess tuples. We use this to show the following property for the excess tuples.

$$\begin{aligned} \exists t' \geq Pexp.TT^{\vdash} \ (&\neg Pvac_{t'-1}(u) \wedge Pvac_{t'}(u) \wedge \\ &\sigma_{[\![P^{R_i}]\!]_{t'}}(R_i \cup R_i^S) = \sigma_{[\![P^{R_i}]\!]_{t'}}((R_i \cup R_i^S) - \{u\})) \end{aligned} \tag{43}$$

This implies that the excess tuples were removed after time $Pexp.TT^{\vdash}$ without $NewDelete$ finding a reason, based on shadow relation predicates P^{R_i}, to store them in the shadow relations. Knowing that, we use induction in the structure of the shadow relation predicates P^{R_i} to prove (39). (In the proof we omit $db_{overhead}$ from the left-hand side of implication (39) due to (42).)

First we show (42). Since $u \in db_{overhead}$ there exist relations R_i and R_i^O and a time $t' < Pexp.TT^{\vdash}$ such that

$$\begin{aligned} u \in ([\![R_i]\!]_{t'-1}, [\![V]\!]_{t'-1}) \wedge u \notin ([\![R_i]\!]_{t'}, [\![V]\!]_{t'}) \wedge \sigma_{[\![P^{R_i}]\!]_{t'}}(R_i \cup R_i^S) \neq \sigma_{[\![P^{R_i}]\!]_{t'}}((R_i \cup R_i^S) - \{u\}) \\ \Rightarrow \ \neg Pvac_{t'-1}(u) \wedge Pvac_{t'}(u) \wedge \sigma_{[\![P^{R_i}]\!]_{t'}}(R_i \cup R_i^S) \neq \sigma_{[\![P^{R_i}]\!]_{t'}}((R_i \cup R_i^S) - \{u\}) \\ \Rightarrow \ u \in [\![db^S]\!]_{t'}, \end{aligned}$$

and since $t' < Pexp.TT^{\vdash} \leq t$ we get that $u \in [\![db^S]\!]_t$.

Then we prove (43). $u \in ([\![db]\!]_t, [\![V]\!]_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash} - 1])$ shows that u is not vacuumed at the time $Pexp.TT^{\vdash}$, i.e., $\nexists t' < Pexp.TT^{\vdash} \ (Pvac_{t'}(u))$. Also, $u \notin [\![db]\!]_t$ implies that the tuple

is vacuumed at some time t', and $u \notin [\![db^S]\!]_t$ implies that it was disposable at that time, i.e., $\exists t' \ (Pvac_{t'}(u) \wedge \sigma_{[\![P^{R_i}]\!]_{t'}}(R_i \cup R_i^S) = \sigma_{[\![P^{R_i}]\!]_{t'}}((R_i \cup R_i^S) - \{u\}))$. In conclusion,

$$\exists t' \geq Pexp.TT^{\vdash} \ (\neg Pvac_{t'-1}(u) \wedge Pvac_{t'}(u) \wedge \sigma_{[\![P^{R_i}]\!]_{t'}}(R_i \cup R_i^S) = \sigma_{[\![P^{R_i}]\!]_{t'}}((R_i \cup R_i^S) - \{u\})).$$

Finally we prove (39), having omitted $db_{overhead}$ from the left-hand side. This will show that omitting any of the excess tuples, u, from the right-hand side of the inclusion in (38) will make no difference for the evaluation of the P-view. Thus, the predicate P^{R_i} defined by the denotational semantics in Section 4 qualifies as a predicate $NewDelete$ can use to maintain base and shadow relations, such that $Pexp$ can be evaluated correctly.

To show (39), we show the following implication for each corresponding pair of relations, R_i and R_i^S, and each tuple u in R_i for which it holds that $u \in ([\![db]\!]_t, [\![V]\!]_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash} - 1]) \wedge u \notin ([\![db]\!]_t \cup [\![db^S]\!]_t)$.

$$\begin{aligned} \exists t' \geq Pexp.TT^{\vdash} \ (&\neg Pvac_{t'-1}(u) \wedge Pvac_{t'}(u) \wedge \\ &\sigma_{[\![P^{R_i}]\!]_{t'}}(R_i \cup R_i^S) = \sigma_{[\![P^{R_i}]\!]_{t'}}((R_i \cup R_i^S) - \{u\})) \\ \Rightarrow\ & Pexp(([\![db]\!]_t, [\![V]\!]_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash} - 1]) \cup db_{overhead}) = \\ & Pexp((([\![db]\!]_t, [\![V]\!]_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash} - 1]) \cup db_{overhead}) - \{u\}) \end{aligned} \tag{44}$$

From property (43) of the excess tuples, we know that this is equivalent to proving (39). Thus, it is equivalent to proving that the predicates P^{R_i} generated using the denotational semantics defined in Section 4 qualify as predicates $NewDelete$ can use to correctly choose to store tuples in db^S.

The implication above is shown using induction in the structure of the denotational semantics, i.e., the structure in Equations (6)–(27). Of the different steps in the induction, we only show the proof of two base steps and two induction steps. All other steps are either simpler or their proofs follow the same pattern as the ones shown.

Note that we assume the existence of r P-views defined before $Pexp$, but the proof of each step also holds for $r = 0$.

Equation 7 - Base step: Assume the P-view, $Pexp$, is defined as $\sigma_p(R)$ and that the P-views $PV_1, \ldots, PV_r$ are defined before $Pexp$. For all relations R_i let $[\![P^{R_i}]\!]_{Pexp.TT^{\vdash}-1} = S_{ds}[\![PV_1 \star \cdots \star PV_r]\!]s_{|R_i}$ which denotes the shadow relation predicate collected for R_i.

As one of the base cases, Equation (7) states $S_{ds}[\![\sigma_p(R_i)]\!]s = s[P^{R_i} \mapsto P^{R_i} \vee (p)]$ which in this case is equivalent to $S_{ds}[\![\sigma_p(R_i)]\!]s = s[P^{R_i} \mapsto [\![P^{R_i}]\!]_{Pexp.TT^{\vdash}-1} \vee p]$ such that $[\![P^{R_i}]\!]_{Pexp.TT^{\vdash}} = [\![P^{R_i}]\!]_{Pexp.TT^{\vdash}-1} \vee p$. Thus,

$$\forall t' \geq Pexp.TT^{\vdash} \ ([\![P^{R_i}]\!]_{t'} = [\![P^{R_i}]\!]_{Pexp.TT^{\vdash}-1} \vee p).$$

Now, the left-hand side of (44) implies that

$$\exists t' \geq Pexp.TT^{\vdash} \ (\sigma_{[\![P^{R_i}]\!]_{Pexp.TT^{\vdash}-1} \vee p}(R_i \cup R_i^S) = \sigma_{[\![P^{R_i}]\!]_{Pexp.TT^{\vdash}-1} \vee p}((R_i \cup R_i^S) - \{u\})),$$

so we know that

$$\begin{aligned} \exists t' \geq Pexp.TT^{\vdash} \ (&\sigma_{[\![P^{R_i}]\!]_{Pexp.TT^{\vdash}-1}}(R_i \cup R_i^S) = \sigma_{[\![P^{R_i}]\!]_{Pexp.TT^{\vdash}-1}}((R_i \cup R_i^S) - \{u\}) \wedge \\ &\sigma_p(R_i \cup R_i^S) = \sigma_p((R_i \cup R_i^S) - \{u\})) \end{aligned}$$

and that $p(u) = \text{false}$. Therefore

$$\sigma_p((\llbracket db \rrbracket_t, \llbracket V \rrbracket_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash} - 1]) \cup db_{overhead}) =$$
$$\sigma_p(((\llbracket db \rrbracket_t, \llbracket V \rrbracket_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash} - 1]) \cup db_{overhead}) - \{u\}),$$

and since $Pexp = \sigma_p(R_i)$ Equation (39) follows.

If no other P-views were defined before $Pexp$, $db_{overhead} = \emptyset$, $\llbracket P^{R_i} \rrbracket_{Pexp.TT^{\vdash}-1} = \text{false}$, and the proof would still hold.

Equation 8 - Base step: Assume the P-view, $Pexp$, is defined as $\text{Agg}_{(L,C,func(A_i))}(R_i)$. As one of the base cases, Equation (8) states that

$$S_{ds}\llbracket \text{Agg}_{(L,C,func(A_i))}(R_i) \rrbracket s =$$
$$\begin{cases} s[P^{R_i} \mapsto (P^{R_i}) \vee (min(A_i))] & \text{if } L = \emptyset \wedge func = \text{“}min\text{”} \\ s[P^{R_i} \mapsto (P^{R_i}) \vee (max(A_i))] & \text{if } L = \emptyset \wedge func = \text{“}max\text{”} \\ s[P^{R_i} \mapsto (P^{R_i}) \vee (min(A_i \text{ by } B_1, \ldots, B_n))] & \text{if } L = \{B_1, \ldots, B_n\} \wedge func = \text{“}min\text{”} \\ s[P^{R_i} \mapsto (P^{R_i}) \vee (max(A_i \text{ by } B_1, \ldots, B_n))] & \text{if } L = \{B_1, \ldots, B_n\} \wedge func = \text{“}max\text{”} \\ s[P^{R_i} \mapsto \text{true}] & \text{otherwise} \end{cases}$$

Again the P-views $PV_1, \ldots, PV_r$ are defined before $Pexp$, such that $\llbracket P^{R_i} \rrbracket_{Pexp.TT^{\vdash}-1} = S_{ds}\llbracket PV_1 \star \cdots \star PV_r \rrbracket s_{|R_i}$ which denotes the shadow relation predicate collected for R_i.

First we assume that $L = \emptyset \wedge func = \text{“}min\text{”}$.

Equation (8) states that $S_{ds}\llbracket \text{Agg}_{(L,C,func(A_i))}(R_i) \rrbracket s = s[P^{R_i} \mapsto P^{R_i} \vee min(A_i)]$, where $min(A_i)$ evaluates to true for a tuple u' *if and only if* $\forall u'' \in R_i \ (u''.A_i \geq u'.A_i)$. This is equivalent to $S_{ds}\llbracket \text{Agg}_{(L,C,func(A_i))}(R_i) \rrbracket s = s[P^{R_i} \mapsto \llbracket P^{R_i} \rrbracket_{Pexp.TT^{\vdash}-1} \vee min(A_i)]$ such that $\llbracket P^{R_i} \rrbracket_{Pexp.TT^{\vdash}} = \llbracket P^{R_i} \rrbracket_{Pexp.TT^{\vdash}-1} \vee min(A_i)$. Thus,

$$\forall t' \geq Pexp.TT^{\vdash} \ (\llbracket P^{R_i} \rrbracket_{t'} = \llbracket P^{R_i} \rrbracket_{Pexp.TT^{\vdash}-1} \vee min(A_i)).$$

Now, the left-hand side of (44) implies that

$$\exists t' \geq Pexp.TT^{\vdash} \ (\sigma_{\llbracket P^{R_i} \rrbracket_{Pexp.TT^{\vdash}-1} \vee min(A_i)}(R_i \cup R_i^S) =$$
$$\sigma_{\llbracket P^{R_i} \rrbracket_{Pexp.TT^{\vdash}-1} \vee min(A_i)}((R_i \cup R_i^S) - \{u\})),$$

so we know that

$$\exists t' \geq Pexp.TT^{\vdash} \ (\sigma_{\llbracket P^{R_i} \rrbracket_{Pexp.TT^{\vdash}-1}}(R_i \cup R_i^S) = \sigma_{\llbracket P^{R_i} \rrbracket_{Pexp.TT^{\vdash}-1}}((R_i \cup R_i^S) - \{u\}) \wedge$$
$$\sigma_{min(A_i)}(R_i \cup R_i^S) = \sigma_{min(A_i)}((R_i \cup R_i^S) - \{u\}))$$

Now, if $\sigma_{min(A_i)}(R_i \cup R_i^S) = \sigma_{min(A_i)}((R_i \cup R_i^S) - \{u\})$ there must exist a tuple u' such that

$$u' \in (R_i \cup R_i^S) \wedge \forall u'' \in (R_i \cup R_i^S)(u''.A_i \geq u'.A_i) \wedge$$
$$u' \in ((R_i \cup R_i^S) - \{u\}) \wedge \forall u''' \in ((R_i \cup R_i^S) - \{u\})(u'''.A_i \geq u'.A_i)$$

To satisfy this, u cannot be the tuple u', i.e., $u' \neq u$ and $\exists u' \in ((R_i \cup R_i^S) - \{u\})\ (u'.A_i \leq u.A_i)$. Therefore

$$\begin{aligned}&\mathrm{Agg}_{(L,C,func(A_i))}((\llbracket db \rrbracket_t, \llbracket V \rrbracket_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash} - 1]) \cup db_{overhead}) \ = \\ &\mathrm{Agg}_{(L,C,func(A_i))}(((\llbracket db \rrbracket_t, \llbracket V \rrbracket_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash} - 1]) \cup db_{overhead}) - \{u\})\end{aligned}$$

and since $Pexp = \mathrm{Agg}_{(L,C,func(A_i))}(R_i)$ Equation (39) follows.

The proofs are similar if $L = \emptyset \wedge func = \text{“}max\text{”}$, if $L = \{B_1, \ldots, B_n\} \wedge func = \text{“}min\text{”}$, or if $L = \{B_1, \ldots, B_n\} \wedge func = \text{“}max\text{”}$. However, if $func \notin \{\text{“}min\text{”}, \text{“}max\text{”}\}$ then (8) states that $S_{ds}\llbracket \mathrm{Agg}_{(L,C,func(A_i))}(R_i) \rrbracket s = s[P^{R_i} \mapsto \mathrm{true}]$ and (44) would imply that

$$\exists t' \geq Pexp.TT^{\vdash}\ (\sigma_{\mathrm{true}}(R_i \cup R_i^S) = \sigma_{\mathrm{true}}((R_i \cup R_i^S) - \{u\}))$$

which is self-contradicting. Thus if such an aggregating P-view exists, no tuples u will exist satisfying $u \in (\llbracket db \rrbracket_t, \llbracket V \rrbracket_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash} - 1]) \wedge u \notin (\llbracket db \rrbracket_t \cup \llbracket db^S \rrbracket_t)$, i.e., there would be no excess tuples to prove disposable.

Again, if no other P-views were defined before $Pexp$, $db_{overhead} = \emptyset$ and $\llbracket P^{R_i} \rrbracket_{Pexp.TT^{\vdash}-1} = \mathrm{false}$, and the proof would still hold.

Equation 13 - Induction step: Assume the P-view, $Pexp$, in question is defined as $Pexp = Y - Z$, and that the P-views $PV_1, \ldots, PV_r$ are defined before $Pexp$. For all relations R_i let $\llbracket P^{R_i} \rrbracket_{Pexp.TT^{\vdash}-1} = S_{ds}\llbracket PV_1 \star \cdots \star PV_r \rrbracket s_{|R_i}$ which denotes the shadow relation predicate collected for R_i.

Then Equation (13) states that $S_{ds}\llbracket Y - Z \rrbracket s = S_{ds}\llbracket Y \star Z \rrbracket s$. Thus, collecting a predicate, P^{R_i}, that allows a correct computation of $Pexp = Y - Z$, can be done as by $S_{ds}\llbracket Y \star Z \rrbracket s$. (We note that the state s at time $Pexp.TT^{\vdash} - 1$ holds the predicates $\llbracket P^{R_i} \rrbracket_{Pexp.TT^{\vdash}-1}$.)

Recall that the $\star$-operator is defined as follows.

$$\begin{aligned}S_{ds}\llbracket Y \star Z \rrbracket s = s' \ \text{ if } \exists s_1, s_2\ ((&S_{ds}\llbracket Y \rrbracket s = s_1 \wedge S_{ds}\llbracket Z \rrbracket s_1 = s') \wedge \\ &(S_{ds}\llbracket Z \rrbracket s = s_2 \wedge S_{ds}\llbracket Y \rrbracket s_2 = s'))\end{aligned}$$

For the P-view Y, if $S_{ds}\llbracket Y \rrbracket s = s_1$ collects predicates $P_{Y_1}^{R_i}$ for all relations R_i in expression Y and if $u \notin (\llbracket db \rrbracket_t \cup \llbracket db^S \rrbracket_t)$, then the induction hypothesis implies the following for all R_i in the expression Y.

$$\begin{aligned}&\exists t' \geq Pexp.TT^{\vdash}\ (\sigma_{\llbracket P_{Y_1}^{R_i} \rrbracket_{t'}}(R_i \cup R_i^S) = \sigma_{\llbracket P_{Y_1}^{R_i} \rrbracket_{t'}}((R_i \cup R_i^S) - \{u\})) \\ &\quad \Rightarrow Y((\llbracket db \rrbracket_t, \llbracket V \rrbracket_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash} - 1]) \cup db_{overhead}) = \\ &\qquad Y(((\llbracket db \rrbracket_t, \llbracket V \rrbracket_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash} - 1]) \cup db_{overhead}) - \{u\})\end{aligned}$$

If $S_{ds}\llbracket Z \rrbracket s_1 = s'$ collects predicates $P_{Z_1}^{R_i}$ for all relations R_i in Z where $P_{Z_1}^{R_i}$ have the same properties as $P_{Y_1}^{R_i}$, a similar expression is implied by the induction hypothesis for the P-view Z. (This also holds if $S_{ds}\llbracket Z \rrbracket s = s_2$ collects predicates $P_{Z_2}^{R_i}$ and $S_{ds}\llbracket Y \rrbracket s_2 = s'$ collects predicates $P_{Y_2}^{R_i}$.)

Now, having $S_{ds}\llbracket Y \star Z \rrbracket s = s'$ where s' for all relations R_i has collected $P^{R_i} = P_{Y_1}^{R_i} \vee P_{Z_1}^{R_i}$ (or $P^{R_i} = P_{Z_2}^{R_i} \vee P_{Y_2}^{R_i}$), then (44) implies that

$$
\begin{aligned}
&\exists t' \geq Pexp.TT^{\vdash}\ (\sigma_{[\![P^{R_i}]\!]_{t'}}(R_i \cup R_i^S) = \sigma_{[\![P^{R_i}]\!]_{t'}}((R_i \cup R_i^S) - \{u\})) \\
&\Rightarrow [Y(([\![db]\!]_t, [\![V]\!]_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash} - 1]) \cup db_{overhead}) \ = \\
&\quad Y((([\![db]\!]_t, [\![V]\!]_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash} - 1]) \cup db_{overhead}) - \{u\})] \wedge \\
&\quad [Z(([\![db]\!]_t, [\![V]\!]_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash} - 1]) \cup db_{overhead}) \ = \\
&\quad Z((([\![db]\!]_t, [\![V]\!]_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash} - 1]) \cup db_{overhead}) - \{u\})].
\end{aligned}
$$

This allows us to conclude

$$
\begin{aligned}
&\exists t' \geq Pexp.TT^{\vdash}\ (\sigma_{[\![P^{R_i}]\!]_{t'}}(R_i \cup R_i^S) = \sigma_{[\![P^{R_i}]\!]_{t'}}((R_i \cup R_i^S) - \{u\})) \\
&\Rightarrow (Y - Z)\big(([\![db]\!]_t, [\![V]\!]_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash} - 1]) \cup db_{overhead}\big) \ = \\
&\quad (Y - Z)\big((([\![db]\!]_t, [\![V]\!]_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash} - 1]) \cup db_{overhead}) - \{u\}\big),
\end{aligned}
$$

i.e., since $Pexp = Y - Z$ Equation (39) follows.

In conclusion, having the right-hand side of Equation (13) collect a P^{R_i} as above, supports maintenance of base and shadow relations such that the P-view can be evaluated correctly.

Equation 19 - Induction step: Assume the P-view, $Pexp = \sigma_{p_{Y\times Z}}(Y \times Z)$, is the P-view in question and that $p_{Y\times Z} = p_Y \wedge p_Z \wedge p_{\{Y,Z\}}$ is in CNF, p_Y being all the predicates that only condition tuples in expression Y and only condition on information from relations taking part in this expression. Similarly p_Z only conditions the expression Z, and $p_{\{Y,Z\}}$ conditions on both at the same time, or conditions on both disjunctively.

Then Equation (19) says that $S_{ds}[\![\sigma_{p_{Y\times Z}}(Y \times Z)]\!]s = S_{ds}[\![\sigma_{p_Y}(Y) \star \sigma_{p_Z}(Z)]\!]s$, i.e., the predicates P^{R_i} collected on the right-hand side will allow a correct evaluation of $Pexp = \sigma_{p_{Y\times Z}}(Y \times Z)$.

As for the other steps, we assume that the P-views $PV_1, \ldots, PV_r$ are defined before $Pexp$, such that $[\![P^{R_i}]\!]_{Pexp.TT^{\vdash}-1} = S_{ds}[\![PV_1 \star \cdots \star PV_r]\!]s_{|R_i}$ which denotes the shadow relation predicate collected for relation R_i where R_i is any relation in $[\![db]\!]_{Pexp.TT^{\vdash}}$. (We note that the current state s at time $Pexp.TT^{\vdash} - 1$ holds the predicates $[\![P^{R_i}]\!]_{Pexp.TT^{\vdash}-1}$.)

We apply the definition of the $\star$-operator as follows.

$$
\begin{aligned}
S_{ds}[\![\sigma_{p_Y}(Y) \star \sigma_{p_Z}(Z)]\!]s = s' \text{ if } \exists s_1, s_2\ &((S_{ds}[\![\sigma_{p_Y}(Y)]\!]s = s_1 \wedge S_{ds}[\![\sigma_{p_Z}(Z)]\!]s_1 = s') \wedge \\
&(S_{ds}[\![\sigma_{p_Z}(Z)]\!]s = s_2 \wedge S_{ds}[\![\sigma_{p_Y}(Y)]\!]s_2 = s'))
\end{aligned}
$$

For the P-view $\sigma_{p_Y}(Y)$, if $S_{ds}[\![\sigma_{p_Y}(Y)]\!]s = s_1$ collects predicates $P_{Y_1}^{R_i}$ for all relations R_i in expression $\sigma_{p_Y}(Y)$ and if $u \notin ([\![db]\!]_t \cup [\![db^S]\!]_t)$, then the induction hypothesis implies the following for all such relations R_i.

$$
\begin{aligned}
&\exists t' \geq Pexp.TT^{\vdash}\ (\sigma_{[\![P_{Y_1}^{R_i}]\!]_{t'}}(R_i \cup R_i^S) = \sigma_{[\![P_{Y_1}^{R_i}]\!]_{t'}}((R_i \cup R_i^S) - \{u\})) \\
&\quad \Rightarrow \sigma_{p_Y}(Y)(([\![db]\!]_t, [\![V]\!]_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash} - 1]) \cup db_{overhead}) \ = \\
&\qquad \sigma_{p_Y}(Y)((([\![db]\!]_t, [\![V]\!]_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash} - 1]) \cup db_{overhead}) - \{u\})
\end{aligned}
$$

If $S_{ds}[\![\sigma_{p_Z}(Z)]\!]s_1 = s'$ collects predicates $P_{Z_1}^{R_i}$ for all relations R_i in $\sigma_{p_Z}(Z)$ where $P_{Z_1}^{R_i}$ have the same properties as $P_{Y_1}^{R_i}$, then the induction hypothesis implies a similar expression for the P-view

$\sigma_{p_Z}(Z)$. (This also holds if $S_{ds}[\![\sigma_{p_Z}(Z)]\!]s = s_2$ collects predicates $P^{R_i}_{Z_2}$ and $S_{ds}[\![\sigma_{p_Y}(Y)]\!]s_2 = s'$ collects predicates $P^{R_i}_{Y_2}$.)

Now, having $S_{ds}[\![\sigma_{p_Y}(Y) \star \sigma_{p_Z}(Z)]\!]s = s'$ where s' for all relations R_i has collected $P^{R_i} = P^{R_i}_{Y_1} \vee P^{R_i}_{Z_1}$ (or $P^{R_i} = P^{R_i}_{Z_2} \vee P^{R_i}_{Y_2}$), then (44) implies that

$$
\begin{aligned}
&\exists t' \geq Pexp.TT^{\vdash}\ (\sigma_{[\![P^{R_i}]\!]_{t'}}(R_i \cup R_i^S) = \sigma_{[\![P^{R_i}]\!]_{t'}}((R_i \cup R_i^S) - \{u\})) \\
&\Rightarrow [\sigma_{p_Y}(Y)(([\![db]\!]_t, [\![V]\!]_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash} - 1]) \cup db_{overhead}) = \\
&\quad \sigma_{p_Y}(Y)((([\![db]\!]_t, [\![V]\!]_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash} - 1]) \cup db_{overhead}) - \{u\})] \wedge \\
&\quad [\sigma_{p_Z}(Z)(([\![db]\!]_t, [\![V]\!]_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash} - 1]) \cup db_{overhead}) = \\
&\quad \sigma_{p_Z}(Z)((([\![db]\!]_t, [\![V]\!]_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash} - 1]) \cup db_{overhead}) - \{u\})]
\end{aligned}
$$

This allows us to conclude

$$
\begin{aligned}
&\exists t' \geq Pexp.TT^{\vdash}\ (\sigma_{[\![P^{R_i}]\!]_{t'}}(R_i \cup R_i^S) = \sigma_{[\![P^{R_i}]\!]_{t'}}((R_i \cup R_i^S) - \{u\})) \\
&\Rightarrow (\sigma_{p_Y}(Y) \times \sigma_{p_Z}(Z))\big(([\![db]\!]_t, [\![V]\!]_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash} - 1]) \cup db_{overhead}\big) = \\
&\quad (\sigma_{p_Y}(Y) \times \sigma_{p_Z}(Z))\big((([\![db]\!]_t, [\![V]\!]_{Pexp.TT^{\vdash}-1}[NOW \leftarrow Pexp.TT^{\vdash} - 1]) \cup db_{overhead}) - \{u\}\big)
\end{aligned}
$$

Since $\sigma_{p_Y}(Y) \times \sigma_{p_Z}(Z) \equiv \sigma_{p_Y \wedge p_Z}(Y \times Z)$ and $\sigma_{p_Y \wedge p_Z \wedge p_{\{Y,Z\}}}(Y \times Z) \subseteq \sigma_{p_Y \wedge p_Z}(Y \times Z)$ Equation (39) follows.

In conclusion, since $Pexp = \sigma_{p_Y \wedge p_Z \wedge p_{\{Y,Z\}}}(Y \times Z)$, having the right-hand side of Equation (19) collect a P^{R_i} as above, and using this to maintain necessary data in the shadow relation creates a dataset that leads to a correct evaluation of the P-view, $Pexp$.

Having proven STEP 1) and STEP 2), Theorem 1 follows. Q.E.D.

B Frequently Used Notation

Symbol	Description
NOW	**Time variable.** A variable evaluating to the current time.
T, T_{NOW}	**Sets of timestamp values.** T is finite and non-empty, and $T_{NOW} = T \cup \{NOW\}$.
$TT^{\vdash}, TT^{\dashv}, VT$, A_i, $Vspec$	**Attribute names.** TT denotes transaction time, and VT denotes valid time; symbols $\vdash$ and $\dashv$ denote the start and end of an interval, respectively. Symbol A_i is used for non-specific attributes, and $Vspec$ is the specific attribute in the vacuuming relation that stores the specification of vacuuming.
$t, t_i, t', t_{now}, t_{def}$	**Timestamps.** t_{now} denotes the current time.
u, u', u_i, v, v', v_i, v_j	**Tuples.** The u's are used for tuples in general, and the v's are used for the tuples that are vacuuming specification parts.
$u.A_i, v.Vspec$, $u.TT^{\vdash}$	**Attribute values.** The ⟨tuple⟩.⟨attribute_name⟩ is generally used to express the value of attribute ⟨attribute_name⟩ for the ⟨tuple⟩.
R, R_i, V	**Relations.** V is the relation storing information on vacuuming specifications.
(R, V)	**Vacuumed relation.** (R, V) denotes the relation R vacuumed by the specification present in V.
db	**Database.**
R^S, db^S	**Shadow structures.** R^S is the shadow relation for relation R, which is used for evaluating P-views on R. db^S denotes the set of shadow relations for the relations in db.
F, F_i	**Selection predicates.** Upper-case letters are used for predicates in vacuuming specifications.
P^R	Shadow relation predicate for relation R. This predicate is used for deciding if a tuple is needed in R^S.
p, p_1, p_R, p_X	Lower-case p's are generally used in the expressions defining P-views.
$[\![u.TT^{\dashv}]\!]_t, [\![R]\!]_t$, $[\![db]\!]_t$	**Evolving structures.** The notation $[\![X]\!]_t$ applies to an evolving structure X, and it returns X as it was at time t. Specifically, $[\![u.TT^{\dashv}]\!]_t$ denotes the value of the transaction time end attribute for tuple u as it was at time t. Likewise, $[\![R]\!]_t$ and $[\![db]\!]_t$ denote R and db as they appeared at time t.
$X[x \leftarrow y]$	**Value replacement.** Denotes X where all occurrences of x are replaced by y.
$s[x \mapsto y]$	**Denotational semantics.** Notation for the state s extended with the assignment of y to x.
$S_{ds}[\![Pexp]\!]s$	Notation for the effect on state s of introducing P-view $Pexp$.

Table 1: Frequently Used Notation