Data Expiration and Aggregate Queries

David Toman

D.R.Cheriton School of Computer Science
University of Waterloo, Canada
david@uwaterloo.ca

Abstract. We investigate the space requirements for summaries needed for maintaining exact answers to aggregate queries over histories of relational databases. We show that, in general, a super-logarithmic lower bound (in the length of the history) on space needed to maintain a summary of the history in order to be able to maintain answers to counting queries. We also develop a natural restriction on the use of aggregation that allows for a logarithmic upper bound and in turn maintaining the summary of the history using counters.

1 Introduction

Data Expiration—the process of removing no longer useful data from database histories while preserving answers to a set of temporal queries—is an essential component of any data warehousing solution that aims on storing and querying information collected over long periods of time. The approaches to data expiration can be evaluated on how well they can remove unnecessary data: the ability of a particular approach to remove data can be quantified in terms of the *size* of the *residual data*, the data that needs to be retained in order to answer queries. The challenge lies in preserving query answers not only at a particular point in time but also for any further extensions of the history: data that may not seem useful at present may become necessary when additional information arrives in the future.

Aggregate queries over histories are often used to summarize the past in a succinct way. Hence, computing aggregates, such as *sum* or *count*, over histories of databases (or data streams) has been a focus of considerable research [5], in particular on maintaining the aggregates over time using as little space as possible.

In this paper we investigate the trade-offs of maintaining *exact* aggregates over database histories. We show that, contrary to common belief that aggregates efficiently summarize (and compress) such histories, the space needed for maintaining exact aggregates is actually larger than that needed for maintaining exact answers to first order queries and, in particular, is *not* bounded by a logarithmic function in the length of the history. This in turn means that exact aggregates cannot be maintained by using counters as conjectured in [8].

The contributions of this paper are as follows:

– We show that unrestricted use of counting in temporal queries leads necessarily to a lower bound of $\Omega(\sqrt{n})$ in the length of a database history; and
– We develop restrictions to the use of the counting aggregate that guarantee that the size of the retained data is bounded by $\mathcal{O}(\log n)$ in the length of the history.

Note that the later restriction still allows for more queries than restricting ourselves to *counting quantifiers*: for counting quantifiers an $\mathcal{O}(1)$ upper bound in the length of the history, the same upper bound that has been shown for temporal relational calculus [8] can be easily achieved by translation to FOL. lso, the later restriction allows unrestricted use of aggregates in queries defined by First-order temporal logic.

The remainder of the paper is organized as follows: Section 2 introduces database histories and temporal queries with aggregation. Section 3 gives a lower bound on the size of the residual data when unrestricted aggregation is allowed. Section 4 develops a restriction on the use of the aggregation that is sufficient to obtain a logarithmic upper bound on the size of the residual data. Section 5 links the results presented in this paper to existing results. Sections:concl concludes with a outline of future directions and open questions.

2 Database Histories and Aggregate Queries

A *database history* records the evolution of a database over time. We assume that time is modeled by positive but not necessarily consecutive integers and we that model the individual consecutive states of the evolution of the database as relational structures. In this setting a finite history of a database is simply a time (integer) indexed sequence of relational database instances:

Definition 1 (Database History) Let ρ be a relational signature. A *database history* H (or a *history* for short) is an integer-indexed sequence of databases $H = (D_0, D_1, \ldots, D_n)$ where D_i is a standard relational database over ρ. We call D_i the i^{th} *state* of H.

The *data domain* $\mathbf{dom}_D$ of a history H is the union of all data values that appear in any relation in D_i at any time instant; the *temporal domain* $\mathbf{dom}_T$ is the set of all time instants that appear as indices in the history H. For a history H we define $\mathsf{MaxT}(H)$ to be the maximal (latest) time instant in $\mathbf{dom}_T$. □

A history H can be *extended* by adding a database D_j, $j > \mathsf{MaxT}(H)$ to the end of the sequence. This process can be repeated arbitrarily many times. Let H' be a sequence of all states successively added to H. We call H' a *suffix* of H and write $H; H'$ for the extension of H by H'.

For the purposes of this paper we restrict our attention only to the *point-based* active domain semantics: the only data values and time instants that exist are those present in the history or are generated by the aggregates. However, we could similarly use, e.g., an *interval based encoding* for the temporal domain

[3] and insist on using consecutive non-overlapping intervals without significant changes in the technical development.

Note also, that the only *update operation* defined for database histories in our framework is the *extension* of the history with a new state. This way our histories can be viewed as *transaction-time* temporal databases. In particular, this arrangement disallows retroactive modifications of the data: retroactive updates negatively impact effectiveness of data expiration [8].

2.1 Aggregate Queries

We use the standard syntax for range-restricted first-order queries extended with a sum aggregate to query database histories.

Definition 2 (Queries) We use the following grammar rule to specify first-order queries with the sum aggregate:

$$Q ::= R(t, \mathbf{x}) \mid \exists x.Q \mid \exists t.Q \mid \mathsf{Agg}_{\mathbf{x}}^{z=\Sigma e}.Q \mid Q \wedge Q \mid Q \wedge \psi \mid Q \wedge \neg Q \mid Q \vee Q$$

where R is a relational symbol, $\mathbf{x}$ is a tuple of variables, t is a temporal variable, and ψ is of the form $x = y$ for data variables and $t = s$ or $t < s$ for temporal variables. $\mathsf{Agg}_{\mathbf{x}}^{z=\Sigma e}.Q$ denotes the sum aggregate operator and e a linear expression. Constants can also be used in the formulas ψ without affecting the result.

We require the queries to obey the standard syntactic safety rules: variables in a condition ψ must appear free in the accompanying query and free variables of subqueries involved in disjunction or negation must match. We also assume that the quantified variables have unique names different from all other variables in the query.

The semantics of the queries is defined using the usual satisfaction relation $\models$ that links histories (D) and substitutions (θ) with queries; the only difference is in the case of base relations: for an $R(\mathbf{x}) \in \rho$ we define $D, \theta \models R(t, \mathbf{x})$ if $\mathbf{x}\theta \in R(D_{t\theta})$. In other words, the atomic predicates are evaluated at the point of the history specified by their first argument.

The $\mathsf{Agg}_{\mathbf{x}}^{z=\Sigma e}.Q$ aggregate operator assigns the variable z the sum of the values of the expression e evaluated with respect to the answer substitutions to Q grouped by the assignments to the variables $\mathbf{x}$.

Without loss of generality we assume that the valuations θ always map variables to values of the appropriate domain and are restricted to the free variables of the particular query. □

We use $\mathsf{Cnt}_{\mathbf{x}}^{z}.Q$ to stand for $\mathsf{Agg}_{\mathbf{x}}^{z=\Sigma 1}.Q$ and to represent the *count* aggregate operator.

2.2 Data Expiration

For historical queries, it is not desirable and often not practical or even feasible to store the entire history in computer storage. Therefore, we devise an *expiration*

operator [8–10] to remember only those parts of the history that are necessary to subsequent answering to queries and extensions of the history. Formally:

Definition 3 (Expiration Operator) Let Q be a query over a history H. An *expiration operator* $\mathcal{E}$ for Q is a triple $(\emptyset, \Delta, \Gamma)$ that satisfies the property

$$Q(H) = \Gamma(\mathcal{E}(H)),$$

where $\mathcal{E}(H)$ is the actual residual data needed to represent the summary of the history we retain in the system and is is defined by:

$$\mathcal{E}(H) = \Delta(D_k, \Delta(D_{k-1}, \Delta(\ldots \Delta(D_0, \emptyset)\ldots))),$$

for every $H = (D_0, D_1, \ldots, D_k)$; $\emptyset$ in this definition is a constant *initial* summary, Δ is a map from summaries and states to summaries, and Γ is a function mapping summaries to answers.. In addition, we require that the triple $(\emptyset, \Delta, \Gamma)$ can be effectively constructed from Q.

The first two components of the expiration operator for Q define a self-maintainable materialized view[1] of H: the $\emptyset$ component tells us what the contents of this view is in the beginning and the Δ component tells us how to update this view when more data arrives in S. The last component, Γ, reproduces the answers to Q while only accessing the information in the view. Note that the definition does not specify what data model the view uses nor what query languages are used for the three components of the operator.

2.3 Properties of Expiration Operators

Intuitively, we have replaced the complete prefix of H with $\mathcal{E}(H)$. Thus our aim is to minimize the size of $\mathcal{E}(H)$ in terms of:

1. the length of the history H, $|\mathbf{dom}_T|$;
2. the number of distinct values in H, $|\mathbf{dom}_D|$; and
3. the size of Q.

The dependency on the length of the history is the most critical factor. In practice, we would like to have $\mathcal{E}(H) \in \mathcal{O}(log|\mathbf{dom}_T|)$, i.e., the size of the residual data is bounded by logarithm of the length of the history—that means that we may need to store, e.g., a counter depending on the length of the history.

The size of the residual data may, however, also depend on the size of the active domain for the data elements, $|\mathbf{dom}_D(t)|$ and the size of the query $|Q|$. This is quite intuitive as the more distinct values are used in the history the more space the residual data is likely take: in most cases we will have to store at least all the uninterpreted constants that appear in H. Similarly, more complex queries are likely to require more data to be retained.

It is easy to see that for queries whose results contain (valuations of) temporal variables can be posed over histories cannot possibly be amenable to data expiry

[1] Not necessarily relational view.

as we may need the information about possibly all the data instants at which the particular history has been updated: this immediately yields a linear lower bound on the size of the residual data with respect to $|\mathbf{dom}_T|$. Hence, for the remainder of the paper we only consider a fixed (number of) queries whose answers only contain data values and possibly aggregate values.

3 Lower Bound for Counting

Consider a history over the schema $\rho = \{p\}$ (a single propositional letter) of the form

$$H = (\emptyset, \underbrace{\{p\}, \ldots, \{p\}}_{m_1}, \emptyset, \underbrace{\{p\}, \ldots, \{p\}}_{m_2}, \emptyset, \ldots \emptyset, \underbrace{\{p\}, \ldots, \{p\}}_{m_k}, \emptyset)$$

where $\emptyset$ stands for an empty database instance (i.e., $\neg p$ holds), $\{p\}$ for a database instance in which p holds (e.g., $H \models P(i)$ for $0 < i \leq m_1$), and $1 < m_i$ for all $0 < i \leq k$ such that $m_i \neq m_j$ for all $0 < i < j \leq k$.

Now consider the query asking the question *"what are the lengths of consecutive p-segments in H"*:

$$\exists t_1, t_2 . t_1 < t_2 \wedge \mathsf{Cnt}^{z}_{\{t_1,t_2\}} . \; (\neg P(t_1) \wedge \neg P(t_2) \wedge (\forall t . t_1 < t < t_2 \rightarrow P(t)) \wedge t_1 < t < t_2 \;)$$

It is easy to see that $Q(H) = \{m_1, m_2, \ldots, m_k\}$. To obtain this answer the residual data has to contain at least

$$\sum_{i=1}^{k} \log m_i = \log \left(\prod_{i=1}^{k} m_i \right) \geq \log 2^k = k$$

bits. Hence for $k = \sqrt{n}$ and $m_1 = i + 1$ we have $|H| \in \mathcal{O}(n)$ but we need at least $\Omega(\sqrt{n})$ bits to represent the residual data for H.

Note also that the need for at least $\Omega(\sqrt{n})$ bits is not necessarily linked to the size of the answer to the query: consider a similar query that asks whether there is a contiguous block of q's of equal length to a block of p's. The later query is boolean, but to be able to provide an answer to this query *for any extension* of H we need to represent at least the counts $m_1, \ldots, m_k$ in the residual history and hence we need $\Omega(\sqrt{n})$ space.

4 Non-splitting Aggregates

The super-logarithmic lower bound presented in section 3 relies crucially on the ability of our query language to count time instants *relatively to other time instants* and this way to generate a large number of distinct aggregate values independently of the size of the data domain $\mathbf{dom}_D$. Indeed, the lower bound presented uses only time-dependent propositions. To avoid this problem, we restrict the use of the aggregation operator $\mathsf{Agg}^{z=\Sigma y}_{\mathbf{x}} . Q$ as follows:

Definition 4 (Non-splitting Aggregate) Let Q be a query and $\mathbf{t}$ be all temporal variables free in Q. We call an aggregate operator $\mathsf{Agg}_{\mathbf{x}}^{z=\Sigma y}.Q$ *non-splitting* if $\mathbf{t} \subseteq \mathbf{x}$ or $\mathbf{t} \cap \mathbf{x} = \emptyset$.

We call a query Q *non-splitting* if all occurrences of aggregate operators in Q are non-splitting.

We extend the approach to data expiration for first-order queries [8] to non-splitting aggregate queries. The technique is based on partial evaluation: we treat relations in the known part of the history H and in all its possible extensions H' as *characteristic* formulas based on equality and order constraints as follows:

Definition 5 (Abstract Substitutions and Formulas) Let H be a history and x and t a data and a temporal variables, respectively, and $\bullet \notin \mathbf{dom}_D \cup \mathbf{dom}_T$ a new symbol; this symbol is used to denote all the values outside of the (current) active data and temporal domains. We define *abstract substitutions* to be the formulas

$$\left[\begin{smallmatrix} x \\ a \end{smallmatrix}\right] \equiv \begin{cases} x = a & a \in \mathbf{dom}_D \\ z = k_{\mathbf{a}} & z \text{ is a result of aggregation} \\ \forall a \in \mathbf{dom}_D . x \neq a & a = \bullet \end{cases}$$

for a data variable where $k_{\mathbf{a}}$ are distinct unknown values (we need at most $|Q| \log^{|Q|} |\mathbf{dom}_D|$ of such values) and

$$\left[\begin{smallmatrix} t \\ s \end{smallmatrix}\right] \equiv \begin{cases} t = s & s \in \mathbf{dom}_T \\ t > \mathsf{MaxT}(\mathbf{dom}_T) & s = \bullet \end{cases}$$

for a temporal variable. We allow composite abstract substitutions to denote a (finite) conjunction of the above formulas (e.g., $\left[\begin{smallmatrix} xy \\ ab \end{smallmatrix}\right]$ denotes the conjunction of $\left[\begin{smallmatrix} x \\ a \end{smallmatrix}\right]$ and $\left[\begin{smallmatrix} y \\ b \end{smallmatrix}\right]$). □

The approach is based on *specializing* a given aggregate query Q with respect to the known part of the history H while keeping the size of the result of the partial evaluation bounded by a function depending only on $|\mathbf{dom}_D|$ and $\log |\mathbf{dom}_T|$. A naive partial evaluation fails as, in the cases of quantification over the temporal domain (and similarly for aggregation over time), the naive replacement of an existential quantifier by a disjunction over all possible time instants would violate this requirement. Hence we devise a equivalence relation $\sim_Q^H$ that, for an existential subquery, classifies the possible *abstract substitutions* to equivalence classes in which all elements *behave the same* with respect to the extensions of the history (i.e., if one is present in an answer after an arbitrary extension of the history, all of them are). This equivalence relation allows choosing a single representative and this way to keep the size of the partially evaluated formula within the required bounds.

The following definition introduces simultaneously both the partial evaluation operation and the equivalence relation:

Definition 6 (Query Specialization and Substitution Equivalence) Let H be a history. We simultaneously define a function PE_H that maps a query Q to a set of *residual queries* indexed by *abstract substitutions* of the form $Q_i[^{\mathbf{x}}_{\mathbf{a}}]$ for $\mathbf{x}$ the set of free variables of Q and $\mathbf{a}$ the corresponding set of abstract values (of the appropriate type), and an equivalence relation $\sim^H_Q$ on abstract substitutions. The function PE_H and the relation $\sim^H_Q$ are defined inductively on the structure of Q as follows:

$R(t, \mathbf{x})$**:** For atomic formulas, the result of partial evaluation with respect to H yields

$$\mathsf{PE}_H(Q) = \{\text{true}[^{t\mathbf{x}}_{s\mathbf{a}}] : R(s, \mathbf{a}) \in D\} \cup \{R(t, \mathbf{x})[^{t\mathbf{x}}_{\bullet\mathbf{a}}] : \mathbf{a} \in (\mathbf{dom}_D \cup \{\bullet\})^{|\mathbf{x}|}\},$$

and the equivalence relation $[^{\mathbf{x}}_{\mathbf{a}_1}] \sim^H_Q [^{\mathbf{x}}_{\mathbf{a}_2}]$ to hold whenever $Q' = Q'' =$ true or $\mathbf{a}_1 = \mathbf{a}_2$ and $s_1 = s_2$ for $Q'[^{t\mathbf{x}}_{s_1\mathbf{a}_1}], Q''[^{t\mathbf{x}}_{s_2\mathbf{a}_2}] \in \mathsf{PE}_H(Q)$;

$Q_1 \wedge F$**:** For a selection, we simply apply the selection on the abstract substitutions:

$$\mathsf{PE}_H(Q) = \{(Q'_1 \wedge F)[^{\mathbf{x}}_{\mathbf{a}}] : [^{\mathbf{x}}_{\mathbf{a}}], Q'_1 \in \mathsf{PE}_H(Q_1), \models [^{\mathbf{x}}_{\mathbf{a}}] \wedge F\}.$$

We set $[^{\mathbf{x}}_{\mathbf{a}_1}] \sim^H_Q [^{\mathbf{x}}_{\mathbf{a}_2}]$ whenever $[^{\mathbf{x}}_{\mathbf{a}_1}] \sim^H_{Q_1} [^{\mathbf{x}}_{\mathbf{a}_2}]$ and $[^{\mathbf{x}}_{\mathbf{a}_1}] \wedge F$ and $[^{\mathbf{x}}_{\mathbf{a}_2}] \wedge F$ are satisfiable;

$Q_1 \wedge Q_2$**:** For a conjunction (join) we combine the abstract substitutions and the residual queries as follows:

$$\mathsf{PE}_H(Q) = \{Q'_1 \wedge Q'_2[^{\mathbf{xy}}_{\mathbf{ab}}] : Q'_1[^{\mathbf{x}}_{\mathbf{a}}] \in \mathsf{PE}_H(Q_1), Q'_2[^{\mathbf{y}}_{\mathbf{b}}] \in \mathsf{PE}_H(Q_2), \models [^{\mathbf{x}}_{\mathbf{a}}] \wedge [^{\mathbf{y}}_{\mathbf{b}}]\}.$$

The equivalence relation $[^{\mathbf{xy}}_{\mathbf{a}_1\mathbf{b}_1}] \sim^H_Q [^{\mathbf{xy}}_{\mathbf{a}_2\mathbf{b}_2}]$ is defined whenever $[^{\mathbf{x}}_{\mathbf{a}_1}] \sim^H_{Q_1} [^{\mathbf{x}}_{\mathbf{a}_2}]$ and $[^{\mathbf{y}}_{\mathbf{b}_1}] \sim^H_{Q_2} [^{\mathbf{y}}_{\mathbf{b}_2}]$ where both $[^{\mathbf{x}}_{\mathbf{a}_1}] \wedge [^{\mathbf{y}}_{\mathbf{b}_1}]$ and $[^{\mathbf{x}}_{\mathbf{a}_2}] \wedge [^{\mathbf{y}}_{\mathbf{b}_2}]$ are satisfiable such that $Q'_1[^{\mathbf{x}}_{\mathbf{a}_1}], Q''_1[^{\mathbf{x}}_{\mathbf{a}_2}] \in \mathsf{PE}_H(Q_1)$ and $Q'_2[^{\mathbf{x}}_{\mathbf{b}_1}], Q''_2[^{\mathbf{x}}_{\mathbf{b}_2}] \in \mathsf{PE}_H(Q_2)$;

$\exists y.Q_1$**:** For an existential subformula quantifying over a data variable we get

$$\mathsf{PE}_H(Q) = \{(\exists y. \bigvee_{Q'_1[^{y\mathbf{x}}_{b\mathbf{a}}] \in \mathsf{PE}_H(Q_1)} Q'_1)[^{\mathbf{x}}_{\mathbf{a}}] : \exists b.Q''[^{y\mathbf{x}}_{b\mathbf{a}}] \in \mathsf{PE}_H(Q_1)\}$$

To define the equivalence relation among the abstract substitutions, let $S_1 = \{b : Q'_1[^{y\mathbf{x}}_{b\mathbf{a}_1}] \in \mathsf{PE}_H(Q')\}$ and $S_2 = \{b : Q'_2[^{y\mathbf{x}}_{b\mathbf{a}_2}] \in \mathsf{PE}_H(Q')\}$. Then $[^{\mathbf{x}}_{\mathbf{a}_1}] \sim^H_Q [^{\mathbf{x}}_{\mathbf{a}_2}]$ whenever for every $b \in S_1$ there is $c \in S_2$ such that $[^{y\mathbf{x}}_{b\mathbf{a}_1}] \sim^H_{Q'} [^{y\mathbf{x}}_{c\mathbf{a}_2}]$ and vice versa.

$\exists t.Q_1$**:** For a temporal existential quantifier we use the equivalence relation as follows: Let $s^{\mathbf{a}}_j$ be a representative of each equivalence class with respect to $\sim^H_{Q_1}$ of all s such that $Q'_1[^{\mathbf{x}t}_{\mathbf{a}s}] \in \mathsf{PE}_H(Q_1)$ (e.g., the smallest value in temporal order). We define

$$\mathsf{PE}_H(Q) = \{(\exists y. \bigvee_{Q'_1[^{\mathbf{x}t}_{\mathbf{a}s^{\mathbf{a}}_j}] \in \mathsf{PE}_H(Q_1)} Q'_1)[^{\mathbf{x}}_{\mathbf{a}}] : \exists s.Q''[^{\mathbf{x}t}_{\mathbf{a}s}] \in \mathsf{PE}_H(Q_1)\}$$

The definition of the equivalence among the resulting abstract substitutions is as above.

$\mathsf{Agg}_{\mathbf{x}}^{z=\Sigma y}\,.\,Q_1$**:** First, consider the case $\mathbf{x} \cap \mathbf{t} = \emptyset$ where $\mathbf{t}$ are all free temporal variables in Q_1. Let $\mathbf{s}_j^{\mathbf{a}b}$ be a representative of each equivalence class with respect to $\sim_{Q_1}^H$ of all $\mathbf{s}$ such that $Q_1'[^{\mathbf{t}\mathbf{x}y}_{\mathbf{s}\mathbf{a}b}] \in \mathsf{PE}_H(Q_1)$ and $k_j^{\mathbf{a}b}$ the cardinality of the class. We define

$$\mathsf{PE}_H(Q) = \{\mathsf{Agg}_{\mathbf{x}}^{z=\Sigma k_j^{\mathbf{a}b}\cdot y}\,.\left(\bigvee_{Q_1'[^{\mathbf{t}\mathbf{x}y}_{\mathbf{s}_j^{\mathbf{a}b}\mathbf{a}b}]\in\mathsf{PE}_H(Q_1)} Q_1'\right)[^{\mathbf{x}z}_{\mathbf{a}k_{\mathbf{a}}}]\},$$

where the value of the aggregate is set to $k_{\mathbf{a}}$, a (unique) unknown value determined by $\mathbf{a}$. Since $\mathbf{x}$ is free of temporal variables and the result of the aggregate is functionally determined by the valuation of $\mathbf{x}$, it is sufficient to define the $\sim_H^Q$ relation to be the diagonal relation on the abstract substitutions.

For the case $\mathbf{t} \subseteq \mathbf{x}$ we have

$$\mathsf{PE}_H(Q) = \{\mathsf{Agg}_{\mathbf{x}}^{z=\Sigma y}\,.\left(\bigvee_{Q_1'[^{\mathbf{x}y}_{\mathbf{a}b}]\in\mathsf{PE}_H(Q_1)} Q_1'\right)[^{\mathbf{x}z}_{\mathbf{a}k_{\mathbf{a}}}]\}.$$

We define the equivalence relation $[^{\mathbf{x}z}_{\mathbf{a}_1k_1}] \sim_Q^H [^{\mathbf{x}z}_{\mathbf{a}_2k_2}]$ to hold whenever for a pair of $\mathbf{a}_1$ and $\mathbf{a}_2$ we can find a matching of the b_1 and b_2 values of y in the abstract answers to Q_1 such that $[^{\mathbf{x}y}_{\mathbf{a}_1b_1}] \sim_Q^H [^{\mathbf{x}y}_{\mathbf{a}_2b_2}]$ holds.

$Q_1 \wedge \neg Q_2$**:** for set difference we define

$$\begin{aligned}\mathsf{PE}_H(Q) = &\{Q_1' \wedge \neg Q_2'[^{\mathbf{x}}_{\mathbf{a}}] : Q_1'[^{\mathbf{x}}_{\mathbf{a}}] \in \mathsf{PE}_H(Q_1), Q_2'[^{\mathbf{x}}_{\mathbf{a}}] \in \mathsf{PE}_H(Q_2)\}\\ &\cup \{Q_1'[^{\mathbf{x}}_{\mathbf{a}}] : Q_1'[^{\mathbf{x}}_{\mathbf{a}}] \in \mathsf{PE}_H(Q_1), Q_2'[^{\mathbf{x}}_{\mathbf{a}}] \notin \mathsf{PE}_H(Q_2)\},\end{aligned}$$

and $[^{\mathbf{x}}_{\mathbf{a}_1}] \sim_Q^H [^{\mathbf{x}}_{\mathbf{a}_2}] \iff \left([^{\mathbf{x}}_{\mathbf{a}_1}] \sim_{Q_1}^H [^{\mathbf{x}}_{\mathbf{a}_2}] \wedge [^{\mathbf{x}}_{\mathbf{a}_1}] \sim_{Q_2}^H [^{\mathbf{x}}_{\mathbf{a}_2}]\right)$, assuming that abstract substitutions not present in $\mathsf{PE}_H(Q_i)$ are related by $\sim_{Q_i}^H$.

$Q_1 \vee Q_2$**:** similarly, for union we have:

$$\begin{aligned}\mathsf{PE}_H(Q) = &\{Q_1' \vee Q_2'[^{\mathbf{x}}_{\mathbf{a}}] : Q_1' \in \mathsf{PE}_H(Q_1)[^{\mathbf{x}}_{\mathbf{a}}], Q_2'[^{\mathbf{x}}_{\mathbf{a}}] \in \mathsf{PE}_H(Q_2)\}\\ &\cup \{Q_1'[^{\mathbf{x}}_{\mathbf{a}}] : Q_1'[^{\mathbf{x}}_{\mathbf{a}}] \in \mathsf{PE}_H(Q_1), Q_2'[^{\mathbf{x}}_{\mathbf{a}}] \notin \mathsf{PE}_H(Q_2)\}\\ &\cup \{Q_2'[^{\mathbf{x}}_{\mathbf{a}}] : Q_1'[^{\mathbf{x}}_{\mathbf{a}}] \notin \mathsf{PE}_H(Q_1), Q_2'[^{\mathbf{x}}_{\mathbf{a}}] \in \mathsf{PE}_H(Q_2)\}\end{aligned}$$

and $[^{\mathbf{x}}_{\mathbf{a}_1}] \sim_Q^H [^{\mathbf{x}}_{\mathbf{a}_2}] \iff \left([^{\mathbf{x}}_{\mathbf{a}_1}] \sim_{Q_1}^H [^{\mathbf{x}}_{\mathbf{a}_2}] \wedge [^{\mathbf{x}}_{\mathbf{a}_1}] \sim_{Q_2}^H [^{\mathbf{x}}_{\mathbf{a}_2}]\right)$, again assuming that abstract substitutions not present in $\mathsf{PE}_H(Q_i)$ are related by $\sim_{Q_i}^H$.

It is easy to prove by induction on the definitions of PE_H and $\sim_Q^H$ that

- $\sim_Q^H$ is an equivalence relation with index bounded by a function of only $|\mathbf{dom}_D|$ and $|Q|$; and
- the formula $\mathsf{PE}_H(Q)$ is bounded in depth by $|Q|$ and with a maximal fan-out (in its syntactic representation) bounded by a function of $|\mathbf{dom}_D|$ and $|Q|$.

The later claim is follows from the former as the disjunctions present in the existential quantifier and in the aggregation depend on the index of $\sim_H$ and not on the size of $\mathbf{dom}_T$ (as they would in a naively partially evaluated Q). As shown in [8] this function is bounded by stack of exponents of height $|Q|$ with a matching lower bound in the case of first-order logic. The additional logarithmic factor $\log|\mathbf{dom}_T|$ comes from the fact that in $\mathsf{PE}_H(Q)$ we need to store the (partial) sums $k_j^{\mathbf{a}b}$ for the aggregates.

4.1 Partial Evaluation based Expiration Operator

The result of $\mathsf{PE}_H(Q)$ can be used directly to encode the the *history* of the history H as follows; to show the second equivalence we extend the PE operator to be able to handle constants in a natural way (omitted in this paper for sake of simplicity):

$$Q(H) = \mathsf{PE}_H(Q)(\emptyset)$$
$$\mathsf{PE}_{H,H'}(Q) \equiv \mathsf{PE}_{H'}(\mathsf{PE}_H(Q))$$

Hence we can simply define the expiration operator to be the triple

$$\langle \mathsf{PE}_\emptyset(Q), \lambda D.\lambda H\, \mathsf{PE}_{\{D\}}(H), \lambda H.H(\emptyset)\rangle.$$

This is sufficient to prove our claims.

Theorem 7 *Let Q be a non-splitting aggregate query. Then*

$$\langle \mathsf{PE}_\emptyset(Q), \lambda D.\lambda H\, \mathsf{PE}_{\{D\}}(H), \lambda H.H(\emptyset)\rangle$$

is a $\log|\mathbf{dom}_T|$-bounded expiration operator for H with respect to Q.

However, in practice, we can extract a subset of H based on collecting the time instants chosen as representatives of the equivalence classes of $\sim_H^Q$ [8]. However, we also need to *assign* the partial counts/sums to these representatives; this can be achieved using a solution to a set of linear equations generated from the cardinalities of the equivalence classes (but is beyond the scope of this paper).

5 Related Work

This work has been inspired by Chomicki's work on bounded history encoding for checking temporal integrity constraints [4]. However, the technique presented in this paper was originally developed for first-order queries [8] and is based on partial evaluation [6]. It is worth mentioning that Chomicki's method for past temporal logic achieves a polynomial upper bound with respect to $\mathbf{dom}_D$ while a similar bound for the method presented in this paper is non elementary: this cannot come as a surprise as first order logic is non-elementarily more succinct than temporal logic (even in the propositional setting).

A parallel stream of research investigates the construction of synopses—summaries of data streams—for the purpose of answering streaming queries [1, 2, 11, 12]. Many of the issues in that setting are common with the approaches to data expiration [10]. However, due to difficulties of maintaining synopses for exact computation of aggregates, a considerable research focused on approximate algorithms [7].

6 Conclusion

We have investigated the space requirements for summaries of database histories needed to maintain exact answers to aggregate queries: the results show that maintaining general aggregates such as count and sum leads to considerable increase in storage requirements over maintaining summaries for first-order queries. We have also developed a syntactic restriction on the use of aggregates that allows the summaries to be bounded by log of the length of the history and in turn implemented using a few additional counters added to a summary that is independent of the length of the history.

Future work should provide a matching upper bound for summaries with respect to general aggregate queries (or to improve on the lower bound presented in this paper). Also we plan to consider alternatives to the naive generation of representatives for the equivalence classes used in the partial evaluation-based technique in order to extract a residual history from the original history H, possibly adorned with values of partial sums and counts.

References

1. Arvind Arasu, Brian Babcock, Shivnath Babu, Jon McAlister, and Jennifer Widom. Characterizing Memory Requirements for Queries over Continuous Data Streams. In *ACM Symposium on Principles of Database Systems*, pages 221–232, 2002.
2. Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and Issues in Data Stream Systems. In *ACM Symposium on Principles of Database Systems*, pages 1–16, 2002.
3. J. Chomicki and D. Toman. Temporal Databases. In M. Fischer, D. Gabbay, and L. Villa, editors, *Handbook of Temporal Reasoning in Artificial Intelligence*, pages 429–467. Elsevier *Foundations of Artificial Intelligence*, 2005.
4. Jan Chomicki. Efficient Checking of Temporal Integrity Constraints Using Bounded History Encoding. *ACM Transactions on Database Systems*, 20(2):149–186, 1995.
5. Graham Cormode and S. Muthukrishnan. An improved data stream summary: The Count-Min sketch and its applications. *Journal of Algorithms*, 55:29–38, 2004.
6. N.D. Jones, C.K. Gomard, and P. Sestoft. *Partial Evaluation and Automatic Program Generation*. Prentice Hall International, 1993.
7. S. Muthukrishnan. *Data Streams: Algorithms and applications*. Now Publishers Inc., 2005.
8. David Toman. Expiration of Historical Databases. In *International Symposium on Temporal Representation and Reasoning*, pages 128–135. IEEE Press, 2001.
9. David Toman. Logical Data Expiration. In Jan Chomicki, Gunter Saake, and Ron van der Meyden, editors, *Logics for Emerging Applications of Databases*, chapter 7, pages 203–238. Springer, 2003.
10. David Toman. On Construction of Holistic Synopses under the Duplicate Semantics of Streaming Queries. In *International Symposium on Temporal Representation and Reasoning*, pages 150–163. IEEE Press, 2007.
11. Jun Yang and Jennifer Widom. Maintaining temporal views over non-temporal information sources for data warehousing. In *Proceedings of EDBT 1998*, pages 389–403, 1998.
12. Jun Yang and Jennifer Widom. Temporal view self-maintenance. In *Proceedings of EDBT 2000*, pages 395–412, 2000.