A Greedy Approach Towards Parsimonious Temporal Aggregation

Juozas Gordevičius Johann Gamper Michael Böhlen
Free University of Bozen-Bolzano, Italy
{gordevicius,gamper,boehlen}@inf.unibz.it

Abstract

Temporal aggregation is a crucial operator in temporal databases and has been studied in various flavors. In instant temporal aggregation (ITA) the aggregate value at time instant t is computed from the tuples that hold at t. ITA considers the distribution of the input data and works at the smallest time granularity, but the result size depends on the input timestamps and can get twice as large as the input relation. In span temporal aggregation (STA) the user specifies the timestamps over which the aggregates are computed and thus controls the result size.

In this paper we introduce a new temporal aggregation operator, called greedy parsimonious temporal aggregation (PTA^g), which combines features from ITA and STA. The operator extends and approximates ITA by greedily merging adjacent tuples with similar aggregate values until the number of result tuples is sufficiently small, which can be controlled by the application. Thus, PTA^g considers the distribution of the data and allows to control the result size. Our empirical evaluation on real world data shows good results: considerable reductions of the result size introduce small errors only.

1 Introduction

Temporal aggregation is a crucial operator in temporal databases that aims to summarize large sets of time-varying information. It has been studied in various flavors, most importantly as instant temporal aggregation (ITA) [2, 3, 4, 6, 7, 8]. ITA works at the smallest time granularity and produces a result tuple whenever an argument tuple starts or ends. Its main drawback is that the result size depends on the argument relation. Due to temporally overlapping argument tuples the result relation is often larger than the argument relation, and can get up to twice as large [2]. This behavior is in conflict with the very idea of aggregation, which is to provide a summary of the data.

Span temporal aggregation (STA) [2, 3] allows to control the result size by permitting an application to specify the time intervals for which to report a result tuple, e.g., for each year from 2000 to 2005. For each such interval a result tuple is produced by aggregating over all argument tuples that overlap the interval. STA is not guaranteed to give good summaries of the data, since the intervals are specified a priori without considering the distribution of the data.

In this paper we present a new data-driven aggregation operator, called *greedy parsimonious temporal aggregation* (PTA^g). It is based on the following observation: many applications do not need the fine-grained result of ITA, but require a concise overview of the data, i.e., a small set of result tuples that represent the most significant changes over time. PTA^g approximates ITA by greedily merging adjacent ITA result tuples that have similar aggregate values until the result relation is sufficiently small. This yields a data-driven approach that allows to control the size of the result relation.

Example 1. Consider a temporal relation PROJECTS that records the name of an employee (E), the project he/she works for (P), his/her monthly salary (S), and a timestamp, T, that represents the time interval (in months) during which the project contract is effective. An instance of the PROJECTS relation is graphically shown in Fig. 1(a), where the timestamps of the tuples are drawn as horizontal lines.

Assume the following ITA query Q1: *"What are the average and maximal monthly salaries for each project?"*. The result of this query is illustrated in Fig. 1(b). It contains more tuples than the input relation, but many tuples represent only small fluctuations in the aggregate values, e.g., $s_1, \ldots, s_5$. To reduce the size of the ITA result, we iteratively merge pairs of the most similar, adjacent tuples. Fig. 1(c) shows a possible reduction to three tuples, which is obtained by applying six consecutive merging steps. Note that the tuples in the reduced result relation show significant changes in the aggregate values.

The similarity between two adjacent tuples is decided upon their aggregate values, and the aggregate values of the new result tuple are computed as the weighted average over the corresponding aggregate values of the merged tuples. For example, the tuples $s_1, \ldots, s_5$ have similar aggregate values and are merged in four steps to produce the PTA^g result tuple z_1, stating that in the interval $[1, 12]$ the average

1530-1311/08 $25.00 © 2008 IEEE
DOI 10.1109/TIME.2008.24

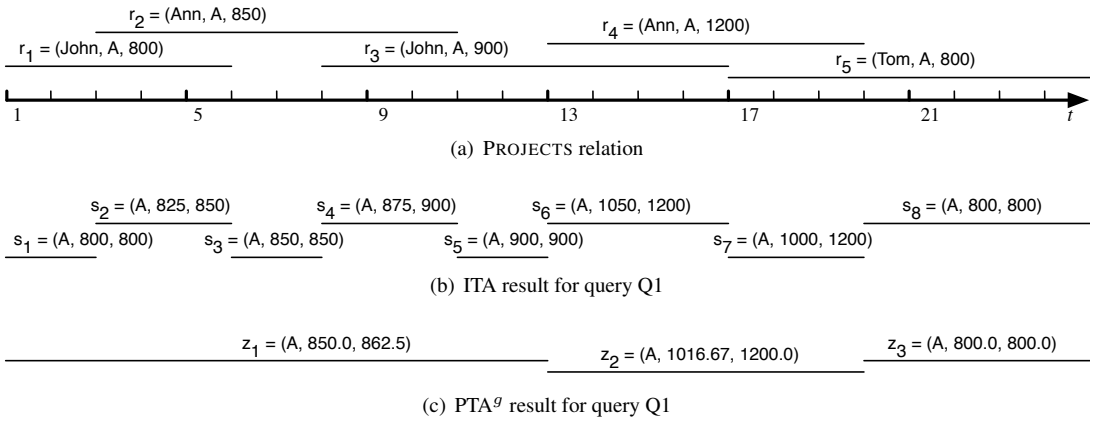


Figure 1. Parsimonious temporal aggregation over the PROJECTS relation.

and maximum salaries for project A are approx. 850 and 862.5, respectively. Note, that in this paper we do not consider merging tuples that are separated with temporal gaps nor tuples from different aggregation groups.

The rest of the paper is organized as follows. Section 2 discusses related work. In Sec. 3 the new temporal aggregation operator is introduced and defined. Section 4 reports about a first empirical evaluation. Section 5 presents conclusions and future work.

2 Related Work

Various forms of temporal aggregation have been proposed, including instant temporal aggregation (ITA), moving-window temporal aggregation, and span temporal aggregation (STA) [2, 3, 4, 5, 6, 7, 8]. They differ mainly in how the time line is partitioned and how the aggregation groups are defined. Most of the past research activities concentrated on the development of efficient evaluation strategies for ITA, both for memory-based evaluation [2, 3, 4] as well as for disk-based evaluation [4, 8].

In [7] temporal aggregation is formalized in a uniform framework that enables the analysis and comparison of the different forms of temporal aggregation. In a similar vein, the multi-dimensional temporal aggregation operator in [2] extends and generalizes previous temporal aggregation operators, by providing more flexibility in partitioning the time line and specifying aggregation groups.

PTA^g is a new temporal aggregation operator that combines features from ITA and STA. By merging similar tuples in the ITA result relation until it is sufficiently small, PTA^g follows a data-driven approach that considers the distribution of the input data and allows to control the size of the result relation.

An approach similar to PTA^g is the approximate temporal coalescing framework in [1]. It is limited to temporal tuples with one numerical attribute and ignores the length of the tuples' timestamp. Adjacent tuples are coalesced if the induced local error is below a user-specified threshold. The PTA^g operator allows multiple attribute values, and the size of the result is controlled by the application.

3 Greedy Parsimonious Temporal Aggregation

In this section we introduce and define a new temporal aggregation operator, termed *greedy parsimonious temporal aggregation* (PTA^g). It takes into consideration the input data distribution and allows to control the size of the result relation. PTA^g approximates ITA and can be defined as a two-step process: (1) compute the ITA result on the argument relation, and (2) iteratively merge pairs of the most similar, adjacent tuples in the ITA result until an application-specific size constraint is satisfied.

Preliminaries. We assume a discrete *time domain*, Δ^T, where the elements are termed chronons (or time points), equipped with a total order, $<^T$ (e.g. calendar months with the order $<$). A timestamp (or time interval) T is a convex set over the time domain and is represented by two chronons, $[TS, TE]$, denoting its inclusive starting and ending points, respectively.

A *relation schema* is a three-tuple $R = (\Omega, \Delta, dom)$, where Ω is a non-empty, finite set of attributes, Δ is a finite set of domains, and $dom : \Omega \rightarrow \Delta$ is a function that associates a domain with each attribute. A *temporal relation schema* is a relation schema with at least one timestamp valued attribute, i.e., $\Delta^T \in \Delta$. A *tuple r over schema R* is a finite set that contains for every $A_i \in \Omega$ a pair A_i/v_i such that $v_i \in dom(A_i)$. A *relation over schema R* is a finite set of tuples over R, denoted as $\mathbf{r}$.

To simplify the notation we assume an ordering of the attributes and represent a temporal relation schema as $R = (A_1, \ldots, A_m, T)$ and a corresponding tuple as $r = (v_1, \ldots, v_m, T)$. For a tuple r and an attribute A we write $r.A$ to denote the value of the attribute A in r. For a set of attributes $A_1, \ldots, A_k$, $k \leq m$, we define $r[A_1, \ldots, A_k] = (r.A_1, \ldots, r.A_k)$.

Instant Temporal Aggregation. For each combination of grouping attribute values, ITA computes an aggregation result at each time point, t, by considering all argument tuples that hold at t and have the same grouping attribute values.

Definition 1. ***(Instant Temporal Aggregation)*** *Let* $\mathbf{r}$ *be a temporal relation with schema* $R = (A_1, \ldots, A_m, T)$, $\mathbf{F} = \{f_{A_{l_1}}, \ldots, f_{A_{l_p}}\}$ *be a set of aggregate functions, and* $\mathbf{A} = \{A_1, \ldots, A_k\}$ *be the grouping attributes. Then the result of* instant temporal aggregation, $\mathbf{s}$, *has the schema* $S = (A_1, \ldots, A_k, f_{A_{l_1}}, \ldots, f_{A_{l_p}}, T)$ *and is defined as*

$$\begin{aligned}\mathcal{G}^{ita}[\mathbf{F}][\mathbf{A}][T]\mathbf{r} = \{x \mid t \in \Delta^T \wedge g \in \pi[\mathbf{A}]\mathbf{r} \wedge \\ \mathbf{r}_g = \{r \mid r \in \mathbf{r} \wedge r[\mathbf{A}] = g \wedge t \in r.T\} \wedge \mathbf{r}_g \neq \emptyset \wedge \\ x = (g.A_1, \ldots, g.A_k, \\ f_{A_{l_1}}(\hat{\pi}[A_{l_1}]\mathbf{r}_g), \ldots, f_{A_{l_p}}(\hat{\pi}[A_{l_p}]\mathbf{r}_g), [t,t])\}\end{aligned}$$

where π *is the projection operator and* $\hat{\pi}$ *is the duplicate preserving projection operator.*

The variable t ranges over the temporal domain, and g ranges over all combinations of grouping attribute values in $\mathbf{r}$. For each combination of t and g, the set $\mathbf{r}_g$ collects the argument tuples that are valid at time t and have the same grouping attribute values as g. If $\mathbf{r}_g$ is not empty, a result tuple, x, is created by extending g with the aggregate values and a timestamp that represents the time instant t. Before applying an aggregate function, $\mathbf{r}_g$ is projected to the attribute the function is applied to. Note that the above definition does not include the final coalescing of value-equivalent tuples over consecutive time points into tuples over maximal time periods during which the aggregate values do not change.

Example 2. The ITA query Q1 over the PROJECTS relation has a single grouping attribute, P, and the aggregate functions are $\mathbf{F} = \{avg(S), max(S)\}$. The final result after coalescing consecutive tuples with identical aggregate values is illustrated in Fig. 1(b).

A fundamental property of ITA aggregation is that the timestamps of the result tuples within a single aggregation group do not intersect. We term such temporal relations sequential.

Definition 2. ***(Sequential Relation)*** *Let* $\mathbf{r}$ *be a temporal relation with schema* $R = (A_1, \ldots, A_k, A_{k+1}, \ldots, A_m, T)$. *Relation* $\mathbf{r}$ *is termed* sequential *with respect to a set of attributes* $\mathbf{A} = \{A_1, \ldots, A_k\}$ *if the following holds true:*

$$\begin{aligned}\forall r_i \forall r_j (r_i \in \mathbf{r} \wedge r_j \in \mathbf{r} \wedge r_i \neq r_j \wedge r_i[\mathbf{A}] = r_j[\mathbf{A}] \\ \implies r_i.T \cap r_j.T = \emptyset)\end{aligned}$$

Example 3. Obviously, the ITA result in Fig. 1(b) is sequential with respect to the grouping attribute P.

Merging Similar Tuples. Having computed the ITA over the argument relation, the next step is to iteratively merge pairs of the most similar, adjacent tuples that have identical grouping attribute values.

Definition 3. ***(Adjacent Tuples)*** *Let* R *and* $\mathbf{A}$ *be as in Def. 2 and* $\mathbf{r}$ *be a sequential relation over* $\mathbf{A}$. *Two tuples* $r_i, r_j \in \mathbf{r}$ *are* adjacent *(with respect to* $\mathbf{A}$*),* $r_i \prec r_j$*, iff*

$$r_i[\mathbf{A}] = r_j[\mathbf{A}] \wedge r_i.TE = r_j.TS - 1$$

Example 4. In Fig. 1(b) we have $s_1 \prec s_2 \prec \cdots \prec s_8$.

When two adjacent tuples, $s_i \prec s_j$, are merged into a new tuple, z, the aggregate values of z are determined as the weighted average, $\oplus$, of the corresponding aggregate values in s_i and s_j. Thus, for an aggregate function, $f_{A_{i_j}}$, the new value in z is computed as $s_i.f_{A_{i_j}} \oplus s_j.f_{A_{i_j}} = \frac{|s_i.T|s_i.f_{A_{i_j}} + |s_j.T|s_j.f_{A_{i_j}}}{|s_i.T|+|s_j.T|}$. Obviously, each merging step introduces a small error with respect to the ITA result. By taking the weighted average of the aggregate values, we get the best approximation of the ITA result (independently of the actual aggregation function).

Now we can define a merge function, m, that takes as input an ITA result relation, $\mathbf{s}$, and a similarity measure, D, and merges the two most similar, adjacent tuples in $\mathbf{s}$.

Definition 4. ***(Merge)*** *Let* $\mathbf{s}$, $\mathbf{A}$, $\mathbf{F}$, *and* T *be as in Def. 1, and* $D : \mathbf{s} \times \mathbf{s} \rightarrow \mathbb{R}^+$ *be a similarity measure. Furthermore, let*

$$\begin{aligned}&(1) \quad (s_i, s_j) = \arg\min_{s \in \mathbf{s}, s' \in \mathbf{s}, s \prec s'} D(s, s') \\ &(2) \quad z = (s.A_1, \ldots, s.A_k, s_i.f_{A_{l_1}} \oplus s_j.f_{A_{l_1}}, \ldots, \\ &\qquad s_i.f_{A_{l_p}} \oplus s_j.f_{A_{l_p}}, [s_i.TS, s_j.TE])\end{aligned}$$

Then the merge *function,* m*, is defined as*

$$m(\mathbf{s}, D) = (\mathbf{s} \setminus \{s_i, s_j\}) \cup \{z\}.$$

s_i and s_j are the two most similar, adjacent tuples in $\mathbf{s}$ according to D. If two or more pairs of tuples have the same smallest distance, the pair s_i, s_j is arbitrarily chosen from these pairs. The new tuple, z, is specified as follows: the values of the grouping attributes are the same as in s_i (or s_j), the timestamp is the period from the starting point of s_i to the ending point of s_j, and the aggregate values are computed as the weighted average.

Example 5. Consider the ITA result in Fig. 1(b), and assume that s_1 and s_2 are the most similar, adjacent tuples according to some D. Then the merge function substitutes s_1 and s_2 by the new tuple $(A, 815.5, 830, [1, 5])$. Merging the latter with s_3 and then with s_4 and s_5 yields the final result tuple z_1 as depicted in Fig. 1(c).

The Similarity Measure. The similarity measure, D, is used to determine pairs of adjacent tuples, $s_i \prec s_j$, with the most similar aggregate values. In this paper we adopt a similarity measure which is based on the weighted distance between the aggregate values of s_i and s_j and the new tuple, z, that would be produced by merging s_i and s_j, i.e., $D(s_i, s_j) =$

$$\sum_{l=l_1}^{l_p}(|s_i.T| \cdot |s_i.f_{A_l} - z.f_{A_l}| + |s_j.T| \cdot |s_j.f_{A_l} - z.f_{A_l}|)$$

where $z.f_{A_l} = s_i.f_{A_l} \oplus s_j.f_{A_l}$. The similarity measure returns 0 when merging two tuples with identical non-temporal attribute values. This is exactly what coalesce does. In all other cases D returns a positive value.

Note that D measures the local error with respect to the ITA result that is introduced when merging two tuples. Other similarity measures might be considered. In particular, it might be useful to introduce weights to leverage the impact of different attributes over different domains to the overall similarity value.

The PTA^g Operator. By applying the merge function iteratively, the ITA result is reduced step-by-step until its size is smaller or equal to a user-defined value c. This leads to the definition of PTA^g.

Definition 5. ***(Greedy Parsimonious Temporal Aggregation)*** *Assume* $\mathbf{r}$, R, $\mathbf{A}$, $\mathbf{F}$ *and* T *as in Def. 1. Further, let* D *be a similarity measure,* m *be the merge function, and* c *be a user-defined value for the result size. Then* greedy parsimonious temporal aggregation *is defined as follows:*

$$\mathcal{G}^{pta}[\mathbf{F}][\mathbf{A}][T][D][c]\mathbf{r} = \hat{m}(\mathcal{G}^{ita}[\mathbf{F}][\mathbf{A}][T]\mathbf{r}, D, c)$$

where

$$\hat{m}(\mathbf{s}, D, c) = \begin{cases} \mathbf{s} & \textit{iff } |\mathbf{s}| \leq c \\ \hat{m}(m(\mathbf{s}, D), D, c) & \textit{otherwise.} \end{cases}$$

Example 6. Figure 1(c) shows the result of PTA^g for Query Q1, where $c = 3$ and D as introduced above.

Lemma 1. *Let* $\mathbf{r}$, $\mathbf{A}$, $\mathbf{F}$, T, E, *and* c *be as in Def. 5. Then the PTA*^{g} *result relation,* $\mathbf{z} = \mathcal{G}^{pta}[\mathbf{F}][\mathbf{A}][T][D][c]\mathbf{r}$*, is a sequential relation.*

Proof. The result of ITA aggregation is a sequential relation and merge function merges only sequential tuples. □

The algorithm for computing PTA^g follows directly the definition. The time complexity is comprised of two parts: the calculation of the ITA and the merge process. In the worst case, the ITA takes $O(n^2)$ time and returns a relation of size $2n - 1$ [2], where n is the size of the argument relation. The merge process over the ITA result has a quadratic worst case complexity $O(n^2)$ when $c = 1$. This complexity stems from the fact that one scan of the ITA result is required to determine the pair with the smallest distance and the number of merges might be $2n-2$. With a heap structure that stores the precomputed distances it is possible to reduce the time complexity of the merging step to $O(n \log n)$.

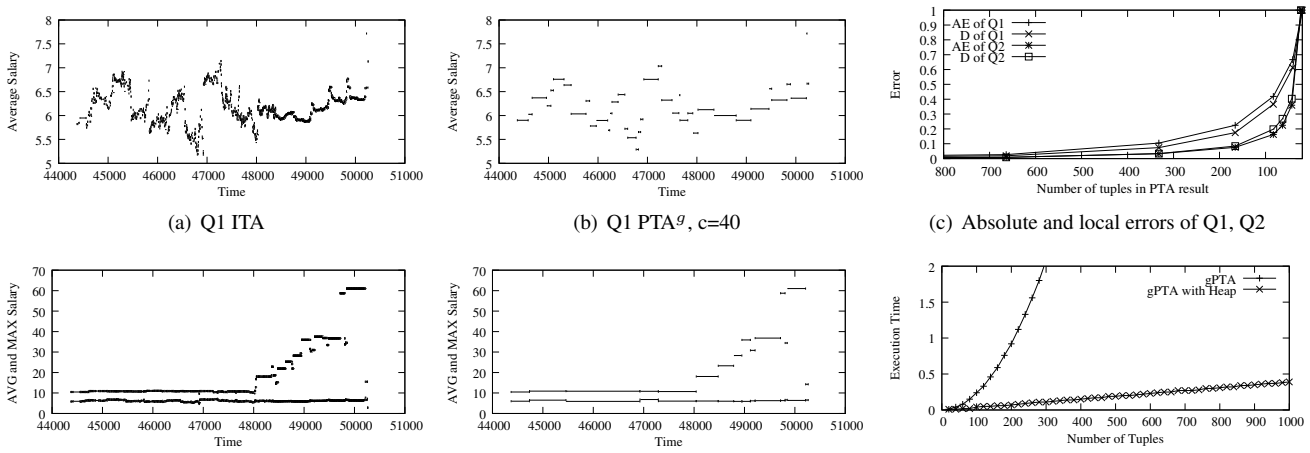
4 Evaluation

We have performed a number of experiments that are mainly focusing on the quality of the PTA^g result using a real-world data set from University of Arizona that stores salary information of employees from 1981 to 1997 at the granularity level of months. The attributes include employee ID, the salary, the department, and the project. The data set contains 83 856 tuples, and the following ITA queries have been issued over this data set:
Q1: *What is the average salary*?
Q2: *What is the average and maximum salary*?
The ITA result of both queries has a total of 2 657 tuples.

Figure 2(a) shows the ITA result of Query Q1, and Fig. 2(b) the corresponding PTA^g result with $c = 40$. It is evident that the reduced result closely resembles the ITA result. Similar, Fig. 2(d) shows the ITA result of Query Q2, and Fig. 2(e) the corresponding PTA^g result with $c = 20$. Here each tuple is represented by two horizontal lines that represent the average and maximum salary, respectively. Again, the resemblance to the ITA result is evident.

To confirm the visual resemblance between the ITA result and the PTA^g result and to measure the quality of the PTA^g result, we compute the absolute error, $AE(\mathbf{s}, \mathbf{z})$, between the ITA and PTA^g result sets. AE is calculated by traversing the time line and summing up at each time point the absolute differences in the aggregate values between the ITA tuples in $\mathbf{s}$ and the corresponding PTA^g tuples in $\mathbf{z}$. (similar as in the distance function, D). We normalize AE with respect to the maximal error, i.e., when c is set to 1.

Fig. 2(c) shows the growth of the absolute error for the two queries. We have observed that the error measure is a good indicator for the optimal termination point of the merging process. During the merging process, the error tends to stay low until some point when it starts to grow up very fast. The best compression to quality ratio is achieved by stopping the merging process at that point. At that point



(a) Q1 ITA (b) Q1 PTA^g, c=40 (c) Absolute and local errors of Q1, Q2

(d) Q2 ITA (e) Q2 PTA^g, c=20 (f) Execution times of PTA^g

Figure 2. Aggregation results, error growth and time complexity of PTA^g operator.

the resources of similar tuples are exhausted, and the algorithm is forced to merge not so similar ones. The same figure plots also the totals of the similarity measure D calculated at each merge step. Observe, that these values grow in the same manner as the absolute error. Thus, we can conclude that the used similarity measure (which actually computes a local error) approximates well the global error.

Fig. 2(f) compares the execution time of our PTA^g implementation. Using the heap offers significant performance improvement.

5 Conclusions and Future Work

In this paper we have introduced a new temporal aggregation operator, called greedy parsimonious temporal aggregation (PTA^g), which combines features from ITA and STA. PTA^g takes the result of ITA and greedily merges adjacent tuples with similar aggregate values to produce a reduced result set, the size of which can be controlled by the user. The new operator takes advantage of the input data distribution and overcomes the main limitation of ITA, where the result set can typically exceed the input in size. A first empirical evaluation shows good results: considerable reductions of the result size introduce small errors only.

The ideas presented in this paper can be extended in various directions. First, we are working on a more efficient evaluation algorithm that computes the PTA^g result directly from the argument relation, thus avoiding the intermediate computation of ITA. Second, the merging process shall be extended to allow merges across different aggregation groups as well as to bridge small temporal gaps between tuples. Third, a careful investigation of different similarity measures is worthwhile. Specifically, when merging across aggregation groups we need a distance function that combines categorical and numerical attributes. Similar, bridging temporal gaps in the merging process will introduce a different kind of error that needs a different treatment. Finally, we will analyze different ways of combining the aggregate values when tuples are merged.

Acknowledgments

We are grateful to the anonymous reviewers for the in-depth reviews of the article and valuable comments.

References

[1] K. Berberich, S. J. Bedathur, T. Neumann, and G. Weikum. A time machine for text search. In *SIGIR*, pages 519–526, 2007.

[2] M. H. Böhlen, J. Gamper, and C. S. Jensen. Multi-dimensional aggregation for temporal data. In *EDBT*, volume 3896 of *LNCS*, pages 257–275. Springer, 2006.

[3] N. Kline and R. T. Snodgrass. Computing temporal aggregates. In *Proceedings of the 11th International Conference on Data Engineering (ICDE'95)*, pages pp. 222–231, Taipei, Taiwan, March 1995.

[4] B. Moon, I. F. Vega Lopez, and V. Immanuel. Efficient algorithms for large-scale temporal aggregation. *IEEE Transactions on Knowledge and Data Engineering*, 15(3):pp. 744–759, May/June 2003.

[5] R. T. Snodgrass, S. Gomez, and L. E. McKenzie. Aggregates in the temporal query language tquel. *IEEE Trans. Knowl. Data Eng.*, 5(5):826–842, 1993.

[6] P. Tuma. *Implementing Historical Aggregates in TempIS*. PhD thesis, Wayne State University, Detroit, Michigan, 1992.

[7] I. F. Vega Lopez, R. T. Snodgrass, and B. Moon. Spatiotemporal aggregate computation: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 17(2):pp. 271–286, 2005.

[8] J. Yang and J. Widom. Incremental computation and maintenance of temporal aggregates. *The VLDB Journal*, 2003.