

**Complex Objects in Data Bases**  
**Multi-dimensional Aggregation of Temporal**  
**Data**

Andreas Bierfert  
December 12, 2006

## Contents

<b>1</b>	<b>Abstract</b>	<b>3</b>
<b>2</b>	<b>Definitions and Remarks</b>	<b>3</b>
2.1	Temporal Data . . . . .	3
2.2	Aggregate Operator . . . . .	3
2.3	Timestamp/Time Interval . . . . .	3
2.4	Relation Schema . . . . .	3
2.4.1	Temporal Relation Schema . . . . .	4
2.5	Attribute Characteristics . . . . .	4
2.6	Adjustment of Attribute Values . . . . .	4
2.7	Constant Intervals . . . . .	5
2.7.1	Cardinality . . . . .	5
2.8	Fixed Intervals . . . . .	5
2.9	Aggregation Factor . . . . .	5
<b>3</b>	<b>Temporal Multi Dimensional Aggregator</b>	<b>5</b>
3.1	Definition . . . . .	5
3.2	Partial Definition of Result Groups . . . . .	6
3.3	Realisation . . . . .	6
3.3.1	Constant Intervals (TMDA-CI) . . . . .	6
3.3.2	TMDA-CI Algorithm . . . . .	7
3.3.3	Fixed Intervals (TMDA-FI) . . . . .	8
3.3.4	TMDA-FI Algorithm . . . . .	8
3.3.5	Complexity . . . . .	8
3.4	Example . . . . .	9
<b>4</b>	<b>Conclusion</b>	<b>10</b>

## 1 Abstract

The paper *Multi-dimensional Aggregation of Temporal Data* is concerned with the evaluation of temporal data in DBMS-systems. With this paper different recent approaches are evaluated and a new operator is introduced which compared with the other approaches solves the problem of temporal data in a efficient way. Following are the most important aspects and definitions needed for the introduction of the operator. The algorithms are then explained in an easy example.

## 2 Definitions and Remarks

This chapter is concerned with definitions and remarks which will lead to the creation of the operator and are required to understand how it works.

### 2.1 Temporal Data

Data is called temporal data if it is in anyway connected to a time interval and by this connection imposes some special aspect of the the data. e.g. that an aspect  $X$  is only valid in the time interval  $[a; b]$ .

### 2.2 Aggregate Operator

The aggregate operator achieves with the partitioning of the argument relation in groups of tuples (with identic values) and the calculation of an aggregate function (e.g **average**) or (**sum**) for the groups a **summary result relation** of the results.

### 2.3 Timestamp/Time Interval

A timestamp (or time interval) is a convex set of **chronons** which are defined as  $[T_s, T_e] =: T$ . The chronons are elements of the discrete timedomain  $D^T$  with the total order  $<^T$ . The following rules apply for the chronos:

- $t \in T$  - chronon  $t$  is element of set  $T$
- $T, T'$  timeintervals,  $T' \subseteq T$  iif.  $\forall t(t \in T \rightarrow t \in T')$
- $T \cap T'$  - chronons which belong to  $T$  and  $T'$
- $T \cap T' \neq \emptyset$  - timeintervals overlap

### 2.4 Relation Schema

A relational schema  $S = (\Omega, \Delta, dom)$  consists of a non-empty set of attributes  $\Omega$ , finit interval sets  $\Delta$  and a function  $dom : \Omega \rightarrow \Delta$ , which connects the interval sets to the attributes.

### 2.4.1 Temporal Relation Schema

A temporal, relational schema includes at least one timeinterval. The timeinterval belongs to  $\Delta$ . From now on we will use the temporal relation schema  $R = (A_1, \dots, A_n, T), G = (B_1, \dots, B_n, T)$ . For better perception we will write the timeinterval at the end of the tuple.

A tuple over schema  $S = (\Omega, \Delta, dom)$  is a function  $r : \rightarrow \cup_{\delta \in \Delta} \delta$ , such that for every attribut holds:  $A \in \Omega, r(A) \in dom(A)$ . We call the tuple temporal iff. the schema it belongs to is temporal. Similar to the schema the order of the tuple is defined as:  $r = (v_1, \dots, v_n, t)$ . A relation over a schema  $R$  is a finite set of tuples denoted with  $\mathbf{r}$ .

For a tuple  $r$  and an attribut  $A$  the following rule applies:  $A$  is attribut of  $r$  then  $r.A$  stands for the value of the attributs. For a finite set of attributes:  $r[A_1, \dots, A_m] = (r.A_1, \dots, r.A_m)$ .

## 2.5 Attribute Characteristics

Three attribute characteristics are defined for temporal data: constant, malleable, atomic. For a schema  $R$  with  $R = (A_1, \dots, A_n, T)$  the characteristics in relation to  $T$  are denoted as:

$C_T = (c_1, \dots, c_n), c_i \in \{c, m, a\}; i \in \{1, \dots, n\}$ .

Example: A schema  $(X, Y, Z, T)$  and attribute characteristics  $C_T = (c, m, a)$  mean that  $X$  is constant,  $Y$  is malleable, and  $Z$  atomic in relation to  $T$ .

## 2.6 Adjustment of Attribute Values

Now that we have the definition of attribute characteristics with a temporal context it is important that the characteristics get evaluated right on a query. Here is a small example to visualize this: Say we have a tuple  $r = (456993, 100, [2006/11/15, 2006/12/15])$ , the first entry being the student identification number, the second the number of hours spent on a project in the time interval denoted by the third component. If we now query on the time interval  $I = [2006/11/15, 2006/12/01]$ , the identification number does not need to be adjusted, the amount of hours spent on the project (if we have an even distribution throughout the whole interval) needs an adjustment. Thus we derive for the tuple  $r$  the attribute characteristics  $C = (c, m)$ . With these characteristics and the according adjustment we get the following result for our query:  $(456993, 50, I)$ .

In general we define an adjustment function  $adj$  for this.

Let  $r = (v_1, \dots, v_n, t)$  be a tuple over schema  $(A_1, \dots, A_n, T)$ , and  $I$  be a time interval and  $C = (c_1, \dots, c_n)$  attribute characteristics then the adjustment function is denoted as follows:

$$adj(r, I, C) = (adj(r.A_1, r.T, I, c_1), \dots, adj(r.A_n, r.T, I, c_n), I)$$

$$adj(v, T, I, c) = \begin{cases} v & c = c' \\ v \cdot |I \cap T| / |T| & c = m' \\ v & c = a' \wedge T = I \\ UNDEF & c = a' \wedge T \neq I \end{cases}$$

## 2.7 Constant Intervals

A constant interval is a maximal, non-overlapping interval in which the set of argument tuples is constant. There are two important things:

- the result intervals must not be greater than the border of the argument intervals
- the result intervals need to be maximal

In a formal definition we get: Let  $r$  be a temporal relation depending on a time interval  $T$ . Then we get the constant intervals of  $r$  with:

$$CI(r) = \{T | \forall r \in r(r.T \supseteq T \vee r.T \cap T = \emptyset) \wedge \forall T' \supset T (\exists r \in r(r.T \not\supseteq T' \wedge r.T \cap T' \neq \emptyset))\}$$

### 2.7.1 Cardinality

The cardinality of a constant interval over a temporal relation  $r$  with  $n$  tuples ( $n > 0$ ) is given by the following formula:

$$|CI(r)| \leq 2n - 1$$

The proof results from the composition of the time intervals of the tuples as they can be linearly ordered (with order  $<^T$ ) thus resulting in a maximum of  $2n$  chronons. As  $n$  chronons can maximally have  $n - 1$  consecutive time intervals we can deduce the formula mentioned above.

## 2.8 Fixed Intervals

We speak about fixed time intervals if overlapping time intervals are in use. Is important that the intervals overlap with the intervals of the argument relation. With this in mind we can influence (with a fitting definition) the cardinality of the result relation. For fixed time intervals we get:

$$\forall T \in FI(r) (\exists r \in r(r.T \cap T \neq \emptyset))$$

## 2.9 Aggregation Factor

The aggregation factor of the aggregate function is nearly the cardinality of the result relation  $z$  and the argument relation  $r$ .

$$af = |z|/|r|$$

# 3 Temporal Multi Dimensional Aggregator

## 3.1 Definition

The most important aspect of the paper is concerned with the introduction of the temporal multi dimensional aggregator (from now on denoted as TMDA). The definition of the result groups is now independent from the input tuples. The definition of the TMDA is as follows:

$\mathbf{r}, \mathbf{g}$  relations with timeinterval  $T$ .  $\mathbf{F} = \{f_{A_{i_1}}, \dots, f_{A_{i_p}}\}$  are aggregate functions

over  $r$ .  $\theta$  is the cardinality of the attributes  $\mathbf{g}$  and  $\mathbf{r}$ .  $C$  are the characteristics of the attributes of  $\mathbf{r}$ . Then holds for the TMDA with the map  $\pi$ , which is used for double entries:

$$G^T[\mathbf{F}][\theta][T][C](\mathbf{g}, \mathbf{r}) =$$

$$\{x | g \in \mathbf{g} \wedge \mathbf{r}_g = \{\{r' | r \in \mathbf{r} \wedge \theta(g, r) \wedge r' = adj(r, g, T, C)\}\}$$

$$\wedge x = g(f_{A_{i_1}}(\pi[A_{i_1}](\mathbf{r}_g)), \dots, f_{A_{i_p}}(\pi[A_{i_p}](\mathbf{r}_g)))\}$$

$r$  here stands for an argument relation and  $g$  a group relation, which constitutes the result groups which are used to calculate the result tuples.  $\theta$  connects an aggregation group  $\mathbf{r}_g \subseteq \mathbf{r}$  with every  $g \in \mathbf{g}$ . The time intervals are adjusted with the *adj*-function according to  $T$ . In the end the aggregation functions are calculated for each aggregation group using a new column for each aggregation.

### 3.2 Partial Definition of Result Groups

TMDA needs completely defined group relations. As the time intervals of the result tuples (in the constant case) get calculated from the argument tuples they are not known in advance. To improve this situation we can calculate the constant time intervals with an easy expression thus leading to a situation where we practically work with fixed intervals:

$$G^T[\mathbf{F}][\theta \wedge overlap(\mathbf{g}, T, \mathbf{r}, T)][T][C](CI(\mathbf{g}', \mathbf{r}, \theta) / \mathbf{g}, r)$$

This sounds really nice but the calculation is really costly as the used operations require quiet intense calculations. Thus the authors of the TMDA came up with partially specified result groups. With this the calculation is done by the TDMA-algorithm

. Upon import of the data relation the calculation is done on-the-fly and the *overlap*-relation is used implicitly.

A result group with schema  $G = (B_1, \dots, B_m, T)$  is partially specified if the value of the time interval is not given. The result tuple is then denoted as  $g = (v_1, \dots, v_m, [*])$ . A conversion into the form of common temporal aggregation operators can be achieved in an easy way.

### 3.3 Realisation

#### 3.3.1 Constant Intervals (TMDA-CI)

With constant intervals an evaluation works as follows: At a chronon  $t$  the values of the argument relation can be used to calculate the result tuple with time intervals ending before  $t$ . Thus new result tuples are generated during computation and only those are kept in memory which are valid in the current time context (**open tuples**). The TMDA-CI algorithm needs five parameters for the calculation of  $G^T$ :

1. group relation  $\mathbf{g}$
2. argument relation  $\mathbf{r}$
3. aggregation functions  $\mathbf{F} = \{f_{A_{i_1}}, \dots, f_{A_{i_p}}\}$

4. decision predicat  $\theta$
5. characteristics  $C$

The algorithm uses two datastructures: A group table  $gt$  with the tuples  $g \in \mathbf{g}$  with a pointer to an end-point-tree  $T$ , and of course the end-point-tree  $T$ , which holds the potential endpoints of the constant intervals.

### 3.3.2 TMDA-CI Algorithm

```

if  $\mathbf{g} = \pi[A_1, \dots, A_m](r)$  then
 $gt \leftarrow$  empty group table with columns  $B_1, \dots, B_m, T, T$ 
else initialize  $gt$  with  $(g, empty\ T), g \in \mathbf{g}$ , and replace timestamp  $T$  by  $[\infty, \cdot]$ ;

```

First the group table is initialized, if  $\mathbf{g}$  is a map over  $\mathbf{r}$ , then the group table is empty and is filled when the argument tuples are read. Otherwise  $gt$  is initialized with  $\mathbf{g}$  by setting the starting times of the entries to  $-\infty$  and generating an empty end-point-tree for each entry.

```

create index for  $gt$  on attributes  $B_1, \dots, B_m; \mathbf{z} \leftarrow \emptyset$ ;

```

Generation of an index over all non-temporal attributes.

```

foreach tuple  $r \in \mathbf{r}$  in chronological order do

```

Now the argument relation  $\mathbf{r}$  is calculated in order of the start times (and  $<^T$ ).

```

if  $\mathbf{g} = \pi[A_1, \dots, A_m](\mathbf{r})$  and  $r.A_1, \dots, r.A_m$  not yet in  $gt$  then
insert  $(r.A_1, \dots, r.A_m, [-\infty, \cdot], empty\ T)$  into  $gt$ ;

```

If the group relations is relational algebra and the expression is not included in the group table yet it will be added.

```

foreach  $i \in Lookup(gt, r, \theta)$  do

```

The *Lookup*-function checks for a data tuple to which result group it belongs. For each fitting result group two things are done:

```

if  $r.T_s > gt[i].T_s$  then
insert a new node with time  $r.T_s - 1$  into  $gt[i].T$  (if not already there);
foreach  $v \in gt[i].T$  in chronological order, where  $v.t < r.T_s$  do
 $gt[i].T_e \leftarrow v.t$ ;
 $\mathbf{z} \leftarrow \mathbf{z} \cup ResultTuple(gt[i], \mathbf{F}, C)$ ;
 $gt[i].T \leftarrow [v.t + 1, \cdot]$ ;
remove node  $v$  from  $gt[i].T$ ;

```

If  $r$  reaches into the current time interval  $r.T_s > gt[i].T_s$ , one or more constant intervals can be closed. Thus  $r.T_s - 1$  denotes a potential endpoint for the constant interval and thus it will be added  $gt[i].T$ . Then all leaves  $v$  of the tree  $gt[i].T$  will be visited, for which the following holds  $v.t < r.T_s$ . The end of the time interval will be set and the result tuple will be generated. After the the

leaf will be deleted from the tree.

$v \leftarrow$  node in  $gt[i].T$  with time  $v.t = r.T_e$  (insert a new node if required);  
 $v.open \leftarrow v.open \cup r[A_1, \dots, A_p, T_s]$ ;

Now the end-point-tree will be refreshed with the new data tuple.

```
foreach  $gt[i] \in gt$  do
  foreach  $v \in gt[i].T$  in chronological order do
    create result tuple, add it to  $\mathbf{z}$ , and close past nodes in  $gt[i].T$ ;
  return  $\mathbf{z}$ 
```

After the calculation the result tuples will be returned in  $\mathbf{z}$

The function *Lookup* searches for a tuple  $r$  of the group table  $gt$  and with the criteria  $\theta$  the fitting result group. For this an AVL-tree is used (with the TMDA-FI even two AVL-trees). The *ResultTuple*-function calculates from the group table  $gt[i]$ , from the aggregation functions  $\mathbf{F}$  and from the characteristics  $C$  the result tuple for the constant interval  $gt[i].T$ . If there are no open tuples in the interval the empty set is returned.

### 3.3.3 Fixed Intervals (TMDA-FI)

For the calculation of  $G^T$  the TMDA-CI uses a tuple  $(\mathbf{g}, \mathbf{r}, \mathbf{F}, \theta, C)$ .  $gt$  is the group table, which includes the group relation  $\mathbf{g}$ .  $\mathbf{g}$  is (compared to the TMDA-CI-algorithm) enhanced by the columns of the aggregation relations  $f_{A_{i_j}} \in \mathbf{F}$ . The result groups are known because of the fixed intervals thus the data tuples do not need to be saved in the end-point-tree.

### 3.3.4 TMDA-FI Algorithm

```
if  $\mathbf{g} = \pi[A_1, \dots, A_m, cast(T, G)](r)$  then
   $gt \leftarrow$  empty group table with columns  $A_1, \dots, A_m, T, f_{A_{i_1}}, \dots, f_{A_{i_p}}$ ;
else
  initialize  $gt$  to  $\mathbf{g}$  and extend it with columns  $f_{A_{i_1}}, \dots, f_{A_{i_p}}$  initialized to NULL;
  Create index for  $gt$  on attribute  $T$ ;
  foreach tuple  $r \in \mathbf{r}$  do
    if  $g = \pi[A_1, \dots, A_m, T](r)$  then
      foreach  $t \in cast(r.T, G)$  do
        Insert  $r.A_1, \dots, r.A_m, t$  into  $gt$  if not already there;
      foreach  $i \in Lookup(gt, r, \theta)$  do
         $r' \leftarrow ADJUST(r, gt[i].T, C)$ ;
      foreach  $f_j \in \mathbf{F}$  do  $gt[i].f_{A_{i_j}} \leftarrow gt[i].f_{A_{i_j}} \oplus r'.A_{i_j}$ ;
  return  $gt$ ;
```

### 3.3.5 Complexity

TMDA-CI

For a complexity analysis of the TMDA-CI only the calculation of the data relation  $\mathbf{r}$  is of interest, which can be split into four important parts:

1. Refreshing of the index
2. *Lookup* of the index
3. Calculation of the result tuples
4. Insertion of the tuples into the end-point-tree

Refreshing and searching of the index as a complexity of  $\log n_g$ ,  $n_g$  denoting the cardinality of the group table. The production of a single result tuple is linear with the number of open tuples  $n_o$ . The amount of result tuples of a data tuple is depending on the aggregation factor  $af = n_z/n_r$ ,  $n_z$  denoting the cardinality of the result relation and  $n_r$  denoting the cardinality of the data relation. Of further interest is the amount of result groups to which  $r$  relates to. This relation can be denoted with  $n_{g,r}$ . The insertion of a tuple into the tree has a complexity of  $\log n_o$ . With this we get a complexity for TMDA-CI of  $O(n_r \max(\log n_g, n_{g,r} af n_o, \log n_i))$ . In the worst-case there exists a case where all start and end point of all intervals are different and all tuples are true thus leading to  $n_o = n_r$  and a complexity of  $O(n_r^2)$ .

#### TMDA-FI

With fixed intervals no open data tuples have to be managed and the values of the aggregation functions can be calculated upon input of the data relation. This leads to a complexity for TMDA-FI of  $O(n_r \max(\log n_g, n_{g,r}))$

### 3.4 Example

For better understanding here is an example from a fictional fire department.

	name	pnr.	department	duration	price	time interval
$r_1$	FF1	1	A	4	7	[11,15]
$r_2$	FF1	1	A	4	8	[18,22]
$r_3$	FF2	2	P	2	12	[12,14]
$r_4$	FF2	2	P	1	12	[14,15]
$r_5$	FF3	3	A	6	9	[17,23]
$r_6$	FF3	3	A	2	7	[13,15]

What we want to solve with TMDA-CI looks like this:

We want to know, at which time intervals the maximum amount of money per department has to be paid.

With  $\prec^T$  we get the following as order of the datasets:

$r_1, r_3, r_6, r_4, r_5, r_2$

For  $gt$  we get with  $r_1$ :

deparment	time interval	$T$
A	[11, ·]	$T_1$
P	$[-\infty, \cdot]$	$T_2$

The end-point-tree has the entry  $15 - \{3, 7, 11\}$  and the result relation  $\mathbf{z}$  is empty. The next tuple is  $r_3$ , which belongs to department  $P$ . As there is one tree per department and calculation is analog to what was just done for the  $A$  department we will only show the interim steps of the  $A$  department. After calculation  $r_6$ ,

$r_4$  is calculated as last datapoint from the  $P$  department. As the time interval of  $r_4$  starts after the end of  $r_3$ , the current open interval will be closed and an result tuple will be generated. The next datapoint is  $r_5$ . As  $r_5$  is outside the open interval the first result tuple for the  $P$  department is calculated and the tree adjusted so that it only contains  $r_5$ . The result relation looks as follows:

department	time interval	sum hours	sum price
A	[13 – 15]	6	14

Running the algorithm till the end gives the following results:

department	time interval	sum hours	sum price
A	[13 – 14]	6	14
A	[17 – 18]	1	9
A	[18 – 22]	8	17
A	[22 – 23]	1	9
P	[12 – 14]	2	12
P	[14 – 15]	1	12

## 4 Conclusion

The TMDA-Operator is an important step in the world of temporal data. There have been, as mentioned in the paper of Böhlen, Gamper und Jensen, a couple of approaches to solve the problems stated here but most where not so efficient or could not deliver results for certain conditions (choise of aggregation functions). Not only is the TMDA-Operator, as shown in the complexity analysis, very efficient it even scales pretty good on test dataset compared to other algorithms. In the future both algorithms TMDA-CI and TMDA-FI can be further improved. The authors of the paper suggest e.g. to initialize the group table on-the-fly or to use an index structure or to experiment with different tree models for the endpointtree. In the end I can say that the introduced operator delivers good possibilities to work with temporal data and leaves enough room for further improvements so that it will even be interesting to use it in the future.

## References

- [1] Multi-dimensional Aggregation for Temporal Data, Michael Böhlen, Johann Gamper, and Christian S. Jensen