How to Aggregate Temporal Data?

Johann Gamper

Free University of Bozen-Bolzano, Italy

Acknowlegements

- *Joint work with A. Dignös, M. Böhlen, J. Gordevicius, G. Mahlknecht*
- *Partially funded by the Autonomous Province of Bozen-Bolzano, Swiss National Science Foundation, and EU (Chorochronos)*

Temporal Data is Ubiquitous

- Almost all information is qualified with time (period or point)
  - Web, RDF stores (triples should really be quintuples)
  - Data warehousing
  - Medical records, loans, ...
  - Sensor data (abstraction from time points into time intervals)
  - Transport information
  - . . .
- Temporal data provide additional and more precise information

From Big Data to Long Data

- Samuel Arbesman: Stop Hyping Big Data and Start Paying Attention to 'Long Data' (Wired 2013)
  - *Sure, big data is a powerful lens for looking at our world ... crunching big numbers can help us learn a lot about ourselves*
  - *But no matter how big that data is or what insights we glean from it, it is still just a snapshot*
  - *We need to stop getting stuck only on big data and start thinking about long data – datasets that have a massive historical sweep*

- Increasing interest in temporal/historical data from big DB vendors

Data Summarization

- Data summarization techniques are needed to
  - reduce these data to an interpretable information for human users
  - reduce the amount of space required to store the data
- Various data summarization technologies/tools exist
  - OLAP (On Line Analytical Processing)
  - Data mining
  - Data visualization
  - (Temporal) aggregation

Outline

1. Different Forms of Temporal Aggregation
2. Computing Temporal Aggregates
3. Parsimonious Temporal Aggregation
4. Systems
5. Conclusions and Future Work

Outline

1. **Different Forms of Temporal Aggregation**
2. Computing Temporal Aggregates
3. Parsimonious Temporal Aggregation
4. Systems
5. Conclusions and Future Work

Non-temporal Aggregation

- Non-temporal aggregation ([Klug'82] and SQL)
  1. Partition relation on the grouping attributes
  2. Compute aggregate functions for each partition
  3. Project to grouping attributes and aggregation results

- **Example:** *Number of contracts per department?*

emp

Name	Dept	T
Jan	DB	[1,12]
Ann	DB	[1,4]
Joe	DB	[5,8]
Tom	AI	[4,9]

Non-temporal Aggregation

- Non-temporal aggregation ([Klug'82] and SQL)
  1. Partition relation on the grouping attributes
  2. Compute aggregate functions for each partition
  3. Project to grouping attributes and aggregation results

- **Example:** *Number of contracts per department?*

emp

Name	Dept	T
Jan	DB	[1,12]
Ann	DB	[1,4]
Joe	DB	[5,8]
Tom	AI	[4,9]

Non-temporal Aggregation

- Non-temporal aggregation ([Klug'82] and SQL)
  1. Partition relation on the grouping attributes
  2. Compute aggregate functions for each partition
  3. Project to grouping attributes and aggregation results

- **Example:** *Number of contracts per department?*

emp

Name	Dept	T	Cnt
Jan	DB	[1,12]	3
Ann	DB	[1,4]	3
Joe	DB	[5,8]	3
Tom	AI	[4,9]	1

Non-temporal Aggregation

- Non-temporal aggregation ([Klug'82] and SQL)
  1. Partition relation on the grouping attributes
  2. Compute aggregate functions for each partition
  3. Project to grouping attributes and aggregation results

- **Example:** *Number of contracts per department?*

emp

Name	Dept	T	Cnt
Jan	DB	[1,12]	3
Ann	DB	[1,4]	3
Joe	DB	[5,8]	3
Tom	AI	[4,9]	1

$\xrightarrow{\pi[Dept, Cnt]\textbf{emp}}$

Dept	Cnt
DB	3
AI	1

Types of Temporal Grouping and Aggregation

- Partition relation along time dimension (+ non-temporal attributes)
- Two types of timeline grouping/partitioning
  - Span grouping: partitioned into fixed-length intervals (months, years)
  - Instant grouping: partitioned into instants/chronons
- Different forms of temporal aggregation
  - Instant temporal aggregation
  - Moving-window temporal aggregation
  - Span temporal aggregation

Instant temporal aggregation (ITA)

- Group tuples for each time point t
  - Aggregate at each t over all tuples valid at t
- Coalesce consecutive tuples with identical aggregate values
  - Lineage information might be considered

- **Example:** Q_{ita} – *Number of employees per department?*

emp

Name	Dept	T
Jan	DB	[1,12]
Ann	DB	[1,4]
Joe	DB	[5,8]
Tom	AI	[4,9]

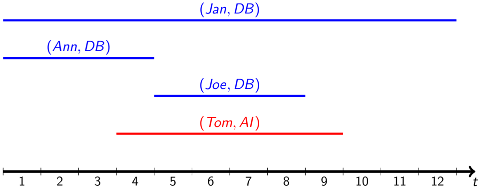
Instant temporal aggregation (ITA)

- Group tuples for each time point t
  - Aggregate at each t over all tuples valid at t
- Coalesce consecutive tuples with identical aggregate values
  - Lineage information might be considered

- **Example:** Q_{ita} – *Number of employees per department?*

emp

Name	Dept	T
Jan	DB	[1,12]
Ann	DB	[1,4]
Joe	DB	[5,8]
Tom	AI	[4,9]



Instant temporal aggregation (ITA)

- Group tuples for each time point t
  - Aggregate at each t over all tuples valid at t
- Coalesce consecutive tuples with identical aggregate values
  - Lineage information might be considered

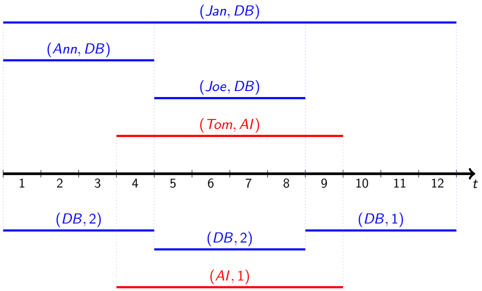
- **Example:** Q_{ita} – *Number of employees per department?*

emp

Name	Dept	T
Jan	DB	[1,12]
Ann	DB	[1,4]
Joe	DB	[5,8]
Tom	AI	[4,9]

Q_{ita}

Dept	Cnt	T
DB	2	[1,4]
DB	2	[5,8]
DB	1	[9,12]
AI	1	[4,9]



Moving-window temporal aggregation (MWTA)

- Group tuples valid in window $[t - w]$ (for each t)
- Coalesce adjacent tuples as in ITA

- **Example:** Q_{mwta} – *Contracts per department in the past 2 months?*

emp

Name	Dept	T
Jan	DB	[1,12]
Ann	DB	[1,4]
Joe	DB	[5,8]
Tom	AI	[4,9]

Q_{mwta}

Dept	Cnt	T
DB	2	[1,4]
DB	3	[5,5]
DB	2	[6,9]
DB	1	[10,13]
AI	1	[4,10]

Moving-window temporal aggregation (MWTA)

- Group tuples valid in window $[t-w]$ (for each t)
- Coalesce adjacent tuples as in ITA

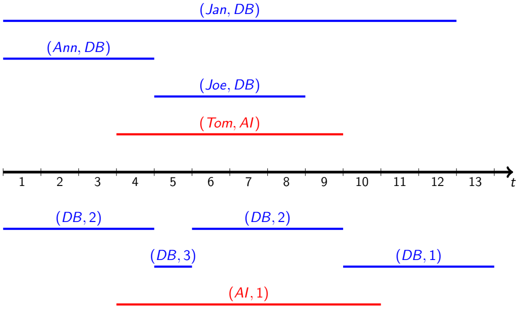
- **Example:** Q_{mwta} – *Contracts per department in the past 2 months?*

emp

Name	Dept	T
Jan	DB	[1,12]
Ann	DB	[1,4]
Joe	DB	[5,8]
Tom	AI	[4,9]

Q_{mwta}

Dept	Cnt	T
DB	2	[1,4]
DB	3	[5,5]
DB	2	[6,9]
DB	1	[10,13]
AI	1	[4,10]



Span temporal aggregation (STA)

- Group tuples valid at time intervals specified in the query

- **Example:** Q_{sta} – *Contracts per department for each semester?*

emp

Name	Dept	T
Jan	DB	[1,12]
Ann	DB	[1,4]
Joe	DB	[5,8]
Tom	AI	[4,9]

Q_{sta}

Dept	Cnt	T
DB	3	[1,6]
DB	2	[7,12]
AI	1	[1,6]
AI	1	[7,12]

Span temporal aggregation (STA)

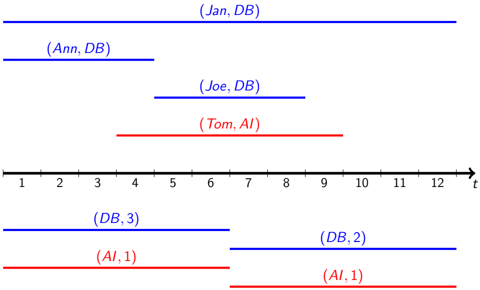
- Group tuples valid at time intervals specified in the query
- **Example:** Q_{sta} – *Contracts per department for each semester?*

emp

Name	Dept	T
Jan	DB	[1,12]
Ann	DB	[1,4]
Joe	DB	[5,8]
Tom	AI	[4,9]

Q_{sta}

Dept	Cnt	T
DB	3	[1,6]
DB	2	[7,12]
AI	1	[1,6]
AI	1	[7,12]



ITA vs. MWTA vs. STA

- ITA and MWTA
  - + Data sensitive, most detailed result
  - – Result typically larger than input relation
  - – Difficult to compute
- STA
  - + Allows to control the result size
  - + Computationally less demanding
  - – Changes over time not well reflected

Outline

1. Different Forms of Temporal Aggregation
2. **Computing Temporal Aggregates**
3. Parsimonious Temporal Aggregation
4. Systems
5. Conclusions and Future Work

Tuma's Work on Temporal Aggregation

- Tuma (1992) proposed a two-step evaluation process for ITA
  1. Compute the constant intervals
  2. Compute the aggregate functions
- Requires two scans of the argument relation!

Aggregation Tree Algorithm [Kline and Snodgrass, ICDE-95]

- Aggregation tree algorithm for ITA
  1. Construct a a tree with time intervals and partial aggregation results
  2. Perform a depth-first traversal on the tree to accumulate the final aggregate values and time intervals
- Requires only one scan of the input relation.
- Avg. complexity: $n \log n$
- Worst case complexity: $O(n^2)$

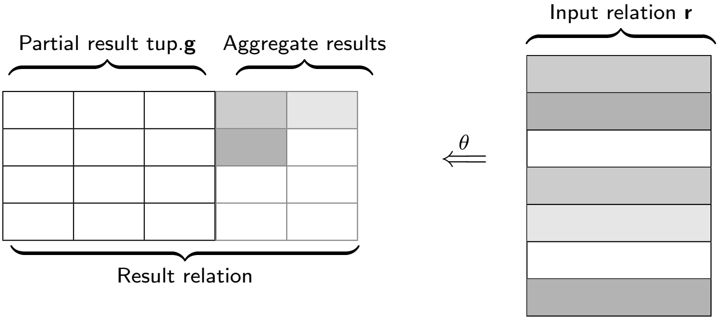
Balanced Tree Algorithm [Moon et al., TKDE-03]

- An algorithm based on timestamp sorting for ITA
  1. Create a (balanced) tree with the start/end time points of the input tuples plus the number of tuples starting/ending at each time point
  2. Traverse the tree in depth-first order and accumulate the values
- Requires only one scan of the input relation
- Complexity: $O(n \log n)$
- Works only for SUM and COUNT, not applicable for MIN and MAX

Temporal Multidimensional Aggregation (TMDA) [Böhlen et al., EDBT-06]

- A general temporal aggregation operator
- Allows the user to specify
  - partial result tuples for which to report aggregation results ($\rightarrow$ **g**)
  - aggregation groups over which to compute the aggregates ($\rightarrow \theta$)

  (both are traditionally determined by the grouping attributes)



TMDA Definition

Let

- $\mathbf{r}$ be an input relation with schema $(A_1, \ldots, A_n)$
- $\mathbf{g}$ be a partial result relation with schema $(B_1, \ldots, B_m)$
- $\theta : \mathbf{r} \rightarrow \mathbf{g}$ be a mapping function
- $\mathbf{F} = \{f_1/C_1, \ldots, f_k/C_k\}$ be a set of aggregation functions

The multidimensional temporal aggregation operator is defined as

$$\begin{aligned} \mathcal{G}^T[\mathbf{g}, \theta, \mathbf{F}]\mathbf{r} = \{g \circ f \mid\ & g \in \mathbf{g} \wedge \\ & \mathbf{r}_g = \{r \in \mathbf{r} \mid \theta(r) = g\} \wedge \\ & f = (f_1(\mathbf{r}_g)/C_1, \ldots, f_k(\mathbf{r}_g)/C_m)\} \end{aligned}$$

The schema of the result relation is $(B_1, \ldots, B_m, C_1, \ldots, C_k, T)$

TMDA Example

- **Example:** Q_{sta} *"Contracts per department for each semester?"*

$$\mathcal{G}^T[\mathbf{g}, \theta, \mathbf{F}]\mathbf{emp}$$

where

$$\mathbf{g} = \pi[Dept, cast(T, semester)]\mathbf{emp} \quad \Longrightarrow$$

$$\theta = (\mathbf{g}.Dept = \mathbf{emp}.Dept \wedge \mathbf{g}.T \cap \mathbf{emp}.T \neq \emptyset)$$

$$\mathbf{F} = \{count(*)/Cnt\}$$

Dept	T	Cnt
DB	[1,6]	
DB	[7,12]	
AI	[1,6]	
AI	[7,12]	

TMDA Example

- **Example:** Q_{sta} *"Contracts per department for each semester?"*

$$\mathcal{G}^T[\mathbf{g}, \theta, \mathbf{F}]\mathbf{emp}$$

where

$$\begin{aligned}\mathbf{g} &= \pi[Dept, cast(T, semester)]\mathbf{emp} \\ \theta &= (\mathbf{g}.Dept = \mathbf{emp}.Dept \wedge \\ &\quad \mathbf{g}.T \cap \mathbf{emp}.T \neq \emptyset) \\ \mathbf{F} &= \{count(*)/Cnt\}\end{aligned} \quad \Longrightarrow$$

Dept	T	Cnt
DB	[1,6]	3
DB	[7,12]	2
AI	[1,6]	1
AI	[7,12]	1

TMDA Properties

- ITA, MTWA, and STA can be expressed with appropriate θ and $\mathbf{g}$
- Partial result relation $\mathbf{g}$ must not depend on $\mathbf{r}$
  - e.g., *For each department in the department relation, . . . ?*
- Use of $\neq, \leq, \geq, \ldots$ operators in θ (in addition to equality)
  - e.g., *For each department, the contracts in the other deptartments?*
    $\theta = g.Dept \neq r.Dept \wedge g.T \cap \mathbf{emp}.T \neq \emptyset$
- Allows efficient computation

TMDA Properties

- ITA, MTWA, and STA can be expressed with appropriate θ and $\mathbf{g}$
- Partial result relation $\mathbf{g}$ must not depend on $\mathbf{r}$
  - e.g., *For each department in the department relation, ...?*
- Use of $\neq, \leq, \geq, \ldots$ operators in θ (in addition to equality)
  - e.g., *For each department, the contracts in the other deptartments?*
    $\theta = g.Dept \neq r.Dept \wedge g.T \cap \mathbf{emp}.T \neq \emptyset$
- Allows efficient computation

TMDA Properties

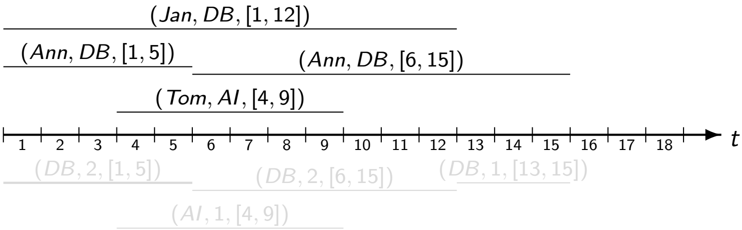
- ITA, MTWA, and STA can be expressed with appropriate θ and $\mathbf{g}$
- Partial result relation $\mathbf{g}$ must not depend on $\mathbf{r}$
  - e.g., *For each department in the department relation, ...?*
- Use of $\neq, \leq, \geq, \ldots$ operators in θ (in addition to equality)
  - e.g., *For each department, the contracts in the other deptartments?*
    $\theta = g.Dept \neq r.Dept \wedge g.T \cap \mathbf{emp}.T \neq \emptyset$
- Allows efficient computation

TMDA Properties

- ITA, MTWA, and STA can be expressed with appropriate θ and $\mathbf{g}$
- Partial result relation $\mathbf{g}$ must not depend on $\mathbf{r}$
  - e.g., *For each department in the department relation, ...?*
- Use of $\neq, \leq, \geq, \ldots$ operators in θ (in addition to equality)
  - e.g., *For each department, the contracts in the other deptartments?*
    $\theta = g.Dept \neq r.Dept \wedge g.T \cap \mathbf{emp}.T \neq \emptyset$
- Allows efficient computation

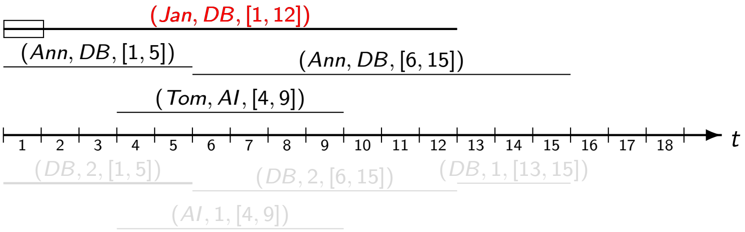
TMDA Algorithm for ITA

- Use a so-called endpoint tree to store "open" tuples
- Scan tuples in chronological order
- At each time point:
  - compute past result tuples
  - update the endpoint tree: add new tuples, remove past tuples



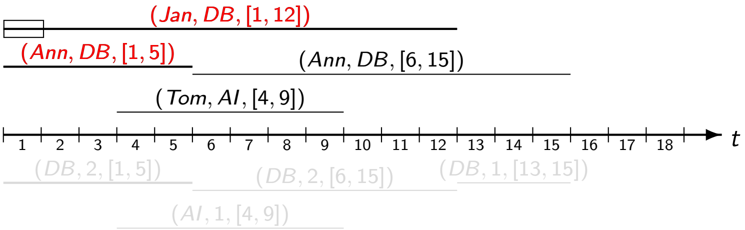
TMDA Algorithm for ITA

- Use a so-called endpoint tree to store “open” tuples
- Scan tuples in chronological order
- At each time point:
  - compute past result tuples
  - update the endpoint tree: add new tuples, remove past tuples



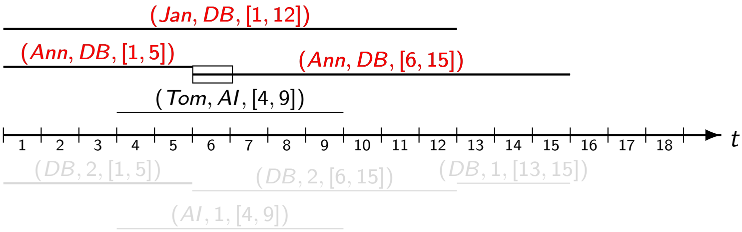
TMDA Algorithm for ITA

- Use a so-called endpoint tree to store "open" tuples
- Scan tuples in chronological order
- At each time point:
  - compute past result tuples
  - update the endpoint tree: add new tuples, remove past tuples



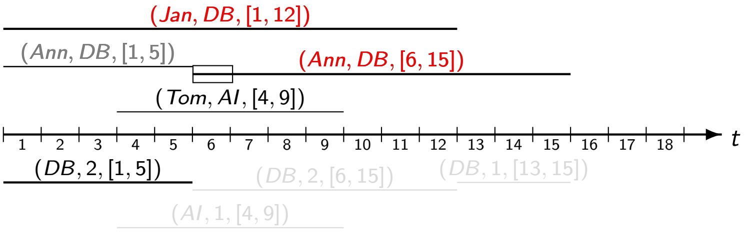
TMDA Algorithm for ITA

- Use a so-called endpoint tree to store "open" tuples
- Scan tuples in chronological order
- At each time point:
  - compute past result tuples
  - update the endpoint tree: add new tuples, remove past tuples



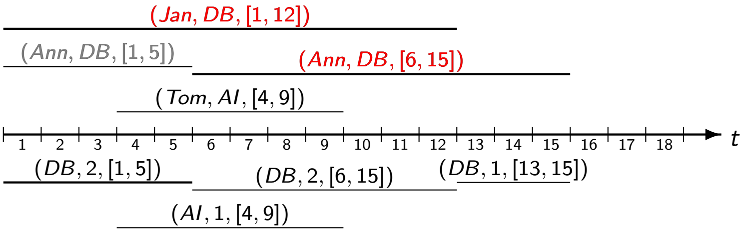
TMDA Algorithm for ITA

- Use a so-called endpoint tree to store "open" tuples
- Scan tuples in chronological order
- At each time point:
  - compute past result tuples
  - update the endpoint tree: add new tuples, remove past tuples



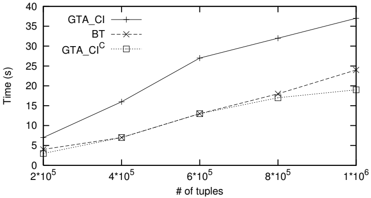
TMDA Algorithm for ITA

- Use a so-called endpoint tree to store "open" tuples
- Scan tuples in chronological order
- At each time point:
  - compute past result tuples
  - update the endpoint tree: add new tuples, remove past tuples

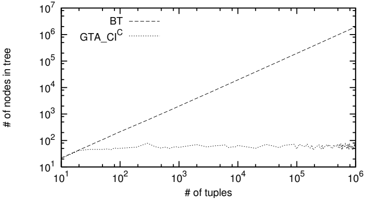


Experimental Evaluation

- TMDA (GTA_CI^c) has the same performance as the Balanced Tree (BT) algorithm [Moon et al., TKDE 2003]
- Memory consumption of GTA_CI is on avg. much less than for BT



Runtime



Space

TMDA Summary

- TMDA is an expressive temporal aggregation operator
  - Allows the specification of partial result tuples and aggregation groups
  - Covers ITA, MWTA, STA
- Efficient evaluation algorithms
  - $O(n \log m)$ (m is number of open tuples)
  - Compares well with state-of-the-art temporal aggregation algorithms
  - Scalable to large data sets

Outline

1. Different Forms of Temporal Aggregation
2. Computing Temporal Aggregates
3. **Parsimonious Temporal Aggregation**
4. Systems
5. Conclusions and Future Work

Parsimonious Temporal Aggregation (PTA)/1 [Gordevicius et al., VLDB Journal 2012]

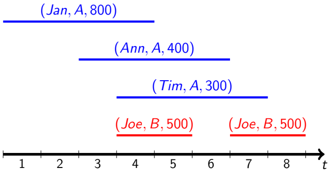
- Overcomes limitations and combines best features of ITA and STA
  - ITA is data-driven/data-sensitive, but the result might be twice as large as input relation
  - STA is not data-driven, but allows to control the result size
- PTA: compress ITA result until a given size/error bound is reached
- Reveals the most significant changes of your data

Parsimonious Temporal Aggregation/2

- **Example:** PTA result of size 4 tuples

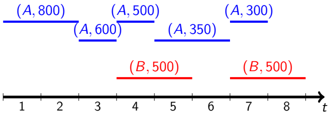
emp

	Emp	Proj	Sal	T
r_1	John	A	800	[1, 4]
r_2	Ann	A	400	[3, 6]
r_3	Tom	A	300	[4, 7]
r_4	John	B	500	[4, 5]
r_5	John	B	500	[7, 8]



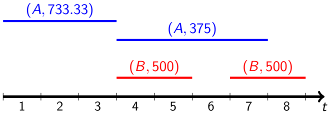
ITA

	Proj	AvgSal	T
s_1	A	800	[1, 2]
s_2	A	600	[3, 3]
s_3	A	500	[4, 4]
s_4	A	350	[5, 6]
		...	



PTA

	Proj	AvgSal	T
z_1	A	733.33	[1, 3]
z_2	A	375	[4, 7]
z_3	B	500	[4, 5]
z_4	B	500	[7, 8]



Computing PTA

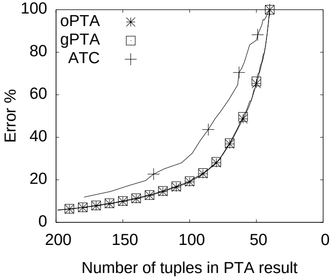
- PTA is an optimization problem
- DP algorithm to compute an optimal solution
  - Explore all possibilities to compress to k tuples
  - ITA must be computed beforehand
  - $O(n^2)$ time and space complexity
  - Optimization: replace DP matrix by a tree structure to reduce space complexity [Mahlknecht et al., Information Systems 2016]
- Greedy algorithm to compute an approximate solution
  - Greedily merge tuples as they are produced by ITA
  - Keep a buffer with $(c + \delta)$ tuples
  - Merge the most similar pair of tuples if buffer overflows
  - $O(n \log(c + \delta))$ time and $O(c + \delta)$ space complexity

Computing PTA

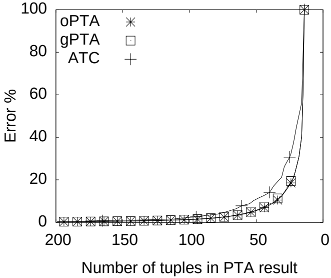
- PTA is an optimization problem
- DP algorithm to compute an optimal solution
  - Explore all possibilities to compress to k tuples
  - ITA must be computed beforehand
  - $O(n^2)$ time and space complexity
  - Optimization: replace DP matrix by a tree structure to reduce space complexity [Mahlknecht et al., Information Systems 2016]
- Greedy algorithm to compute an approximate solution
  - Greedily merge tuples as they are produced by ITA
  - Keep a buffer with $(c + \delta)$ tuples
  - Merge the most similar pair of tuples if buffer overflows
  - $O(n \log(c + \delta))$ time and $O(c + \delta)$ space complexity

Experimental Evaluation

- Considerable reduction of result size introduce only a small error
- Greedy algorithm is very close to the optimal solution and more efficient than competitors



ITA size = 11.643



ITA size = 5.552

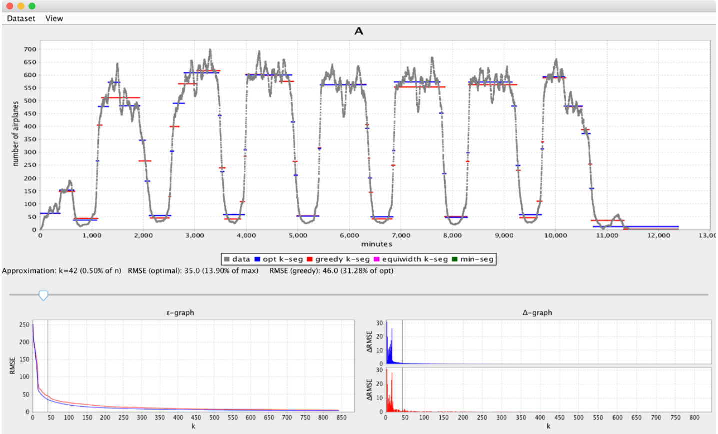
Salary data of UofA

Outline

1. Different Forms of Temporal Aggregation
2. Computing Temporal Aggregates
3. Parsimonious Temporal Aggregation
4. **Systems**
5. Conclusions and Future Work

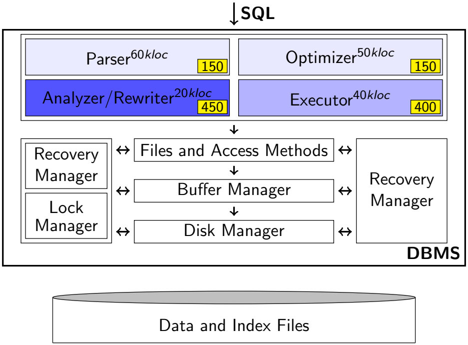
Data Exploration

- Includes our PTA algorithms to explore large data sets



Temporal PostgreSQL

- We have implemented a temporal PostgreSQL (http://tpg.inf.unibz.it/)
- Full-fledged support for all temporal operators, including temporal aggregation



Outline

1. Different Forms of Temporal Aggregation
2. Computing Temporal Aggregates
3. Parsimonious Temporal Aggregation
4. Systems
5. **Conclusions and Future Work**

Conclusions

- Different forms of temporal aggregation: ITA, MWTA, STA
- TMDA is an expressive temporal aggregation operator
- PTA combines the best features of ITA and STA
- Stand-alone algorithmic solutions

Some Future Directions

- Address other aspects of temporal aggregation
  - Two or more time dimensions, e.g., transaction and valid time
  - Streaming/time series data
  - Other application contexts, e.g., IR
- Integrate TMDA and PTA into our temporal PostgreSQL

Thank you!