Temporal Alignment

Anton Dignös[1] Michael H. Böhlen[1] Johann Gamper[2]

[1]University of Zürich, Switzerland

[2]Free University of Bozen-Bolzano, Italy

SIGMOD 2012

May 24, 2012 - Scottsdale, Arizona, USA

Outline

Goal and Problem Definition

Temporal Primitives

Properties of Temporal RA

Implementation and Empirical Evaluation

Related Work

Summary and Future Work

Temporal Data Example

- **Input**: Employee N works for department D during time T.

R

	N	D	T
r_1	Joe	DB	[Feb, Jul)
r_2	Ann	DB	[Feb, Sep)
r_3	Sam	AI	[May, Oct)

- **Query**: How did the average duration of contracts per department change?

- **Result**: Temporal Aggregation: ${}_{D}\vartheta^{T}_{AVG(DUR(T))}(\mathbf{R})$

	AVG	D	T
z_1	6	DB	[Feb, Jul)
z_2	7	DB	[Jul, Sep)
z_3	5	AI	[May, Oct)

Timestamps must be adjusted for the result.

Temporal Data Example

- **Input**: Employee N works for department D during time T.

R

	N	D	T
r_1	Joe	DB	[Feb, Jul)
r_2	Ann	DB	[Feb, Sep)
r_3	Sam	AI	[May, Oct)

- **Query**: How did the average duration of contracts per department change?

- **Result**: Temporal Aggregation: ${}_D\vartheta^T_{AVG(DUR(T))}(\mathbf{R})$

	AVG	D	T
z_1	6	DB	[Feb, Jul)
z_2	7	DB	[Jul, Sep)
z_3	5	AI	[May, Oct)

Timestamps must be adjusted for the result.

Temporal Data Example

- **Input**: Employee N works for department D during time T.

R

	N	D	T
r_1	Joe	DB	[Feb, Jul)
r_2	Ann	DB	[Feb, Sep)
r_3	Sam	AI	[May, Oct)

- **Query**: How did the average duration of contracts per department change?

- **Result**: Temporal Aggregation: $_{D}\vartheta^{T}_{AVG(DUR(T))}(\mathbf{R})$

	AVG	D	T
z_1	6	DB	[Feb, Jul)
z_2	7	DB	[Jul, Sep)
z_3	5	AI	[May, Oct)

Timestamps must be adjusted for the result.

Requirements for Query Processing

- A temporal query must be **reducible to a nontemporal query**.
  - A temporal query is defined by its corresponding nontemporal query.
  - ${}_{D}\vartheta^{T}_{AVG} \cdots \Rightarrow {}_{D}\vartheta_{AVG} \cdots$
- **Original timestamps** have to be **accessible**.
  - Despite timestamp adjustment original timestamps are accessible.
  - ${}_{D}\vartheta^{T}_{AVG(DUR(T))}(\mathbf{R})$
- The **boundaries of timestamps** have to be **preserved**.
  - Timestamps can not be split and/or merged arbitrarily.
  - $\{(\text{DB}, 800\text{k}, [\texttt{Feb}, \texttt{Jul}))\} \neq$
    $\{(\text{DB}, 800\text{k}, [\texttt{Feb}, \texttt{Apr})), (\text{DB}, 800\text{k}, [\texttt{Apr}, \texttt{Jul}))\}$

These are the requirements of the **sequenced semantics**.

Requirements for Query Processing

- A temporal query must be **reducible to a nontemporal query**.
  - A temporal query is defined by its corresponding nontemporal query.
  - $_D\vartheta^T_{AVG} \cdots \Rightarrow {}_D\vartheta_{AVG} \cdots$
- **Original timestamps** have to be **accessible**.
  - Despite timestamp adjustment original timestamps are accessible.
  - $_D\vartheta^T_{AVG(DUR(T))}(\mathbf{R})$
- The **boundaries of timestamps** have to be **preserved**.
  - Timestamps can not be split and/or merged arbitrarily.
  - $\{(\text{DB}, 800\text{k}, [\texttt{Feb}, \texttt{Jul}))\} \neq$
    $\{(\text{DB}, 800\text{k}, [\texttt{Feb}, \texttt{Apr})), (\text{DB}, 800\text{k}, [\texttt{Apr}, \texttt{Jul}))\}$

These are the requirements of the **sequenced semantics**.

Requirements for Query Processing

- A temporal query must be **reducible to a nontemporal query**.
  - A temporal query is defined by its corresponding nontemporal query.
  - ${}_D\vartheta^T_{AVG} \ldots \Rightarrow {}_D\vartheta_{AVG} \ldots$
- **Original timestamps** have to be **accessible**.
  - Despite timestamp adjustment original timestamps are accessible.
  - ${}_D\vartheta^T_{AVG(DUR(T))}(\mathbf{R})$
- The **boundaries of timestamps** have to be **preserved**.
  - Timestamps can not be split and/or merged arbitrarily.
  - $\{(\text{DB}, 800\text{k}, [\texttt{Feb}, \texttt{Jul}))\} \neq$
    $\{(\text{DB}, 800\text{k}, [\texttt{Feb}, \texttt{Apr})), (\text{DB}, 800\text{k}, [\texttt{Apr}, \texttt{Jul}))\}$

These are the requirements of the **sequenced semantics**.

Goal and Problem Definition

Goal: Reduction of sequenced algebra to nontemporal algebra with the help of timestamp adjustment.

Problem Definition: Given a temporal operator ψ^T of the sequenced semantics, and input relations $\mathbf{r}_1, \dots \mathbf{r}_n$, our goal is to express $\psi^T(\mathbf{r}_1, \dots \mathbf{r}_n)$ as follows:

$$\psi^T\Big(\mathbf{r}_1, \dots \mathbf{r}_n\Big) = \psi\Big(\mathcal{P}^T(\mathbf{r}_1, \dots \mathbf{r}_n), \dots \mathcal{P}^T(\mathbf{r}_n, \dots \mathbf{r}_1)\Big) \qquad \textbf{(reduction)}$$

where ψ is the nontemporal operator corresponding to ψ^T, and $\mathcal{P}^T(\mathbf{r}_1, \dots \mathbf{r}_n)$ adjusts the timestamps of $\mathbf{r}_1$.

Goal and Problem Definition

<u>Goal</u>: Reduction of sequenced algebra to nontemporal algebra with the help of timestamp adjustment.

<u>Problem Definition</u>: Given a temporal operator ψ^T of the sequenced semantics, and input relations $\mathbf{r}_1, \ldots \mathbf{r}_n$, our goal is to express $\psi^T(\mathbf{r}_1, \ldots \mathbf{r}_n)$ as follows:

$$\psi^T\Big(\mathbf{r}_1, \ldots \mathbf{r}_n\Big) = \psi\Big(\mathcal{P}^{\mathcal{T}}(\mathbf{r}_1, \ldots \mathbf{r}_n), \ldots \mathcal{P}^{\mathcal{T}}(\mathbf{r}_n, \ldots \mathbf{r}_1)\Big) \qquad \textbf{(reduction)}$$

where ψ is the nontemporal operator corresponding to ψ^T, and $\mathcal{P}^{\mathcal{T}}(\mathbf{r}_1, \ldots \mathbf{r}_n)$ adjusts the timestamps of $\mathbf{r}_1$.

Solution

- **Two new algebra operators** (primitives) for **adjustment** of timestamps:
  - Temporal Splitter $\mathcal{N}$
  - Temporal Aligner ϕ
- Adjustment must allow to propagate original timstamps.
- Adjustment must respect the lineage.
- **Reduction rules** from temporal RA to nontemporal RA.
- **Timestamp propagation** for accessing original timestamps.

Temporal Primitives

- The purpose of a temporal primitive is to break timestamps into pieces.
- Two temporal primitives are required:
  - One input tuple contributes to **at most one** result tuple per time point.
    $\Rightarrow$ **Temporal Splitter**
    Example: Aggregation
  - One input tuple contributes to **more than one** result tuple per time point.
    $\Rightarrow$ **Temporal Aligner**
    Example: Joins

Temporal Splitter

- Average duration of contracts per department: ${}_{D}\vartheta^{T}_{AVG(DUR(T))}(\mathbf{R})$

R

	N	D	T
r_1	Joe	DB	[Feb, Jul)
r_2	Ann	DB	[Feb, Sep)
r_3	Sam	AI	[May, Oct)

adjustment (disjoint)

N	D	U	T
Joe	DB	[Feb, Jul)	[Feb, Jul)
Ann	DB	[Feb, Sep)	[Feb, Jul)

N	D	U	T
Ann	DB	[Feb, Sep)	[Jul, Sep)

N	D	U	T
Sam	AI	[May, Oct)	[May, Oct)

nontemporal aggregation

AVG	D	T
6	DB	[Feb, Jul)

AVG	D	T
7	DB	[Jul, Sep)

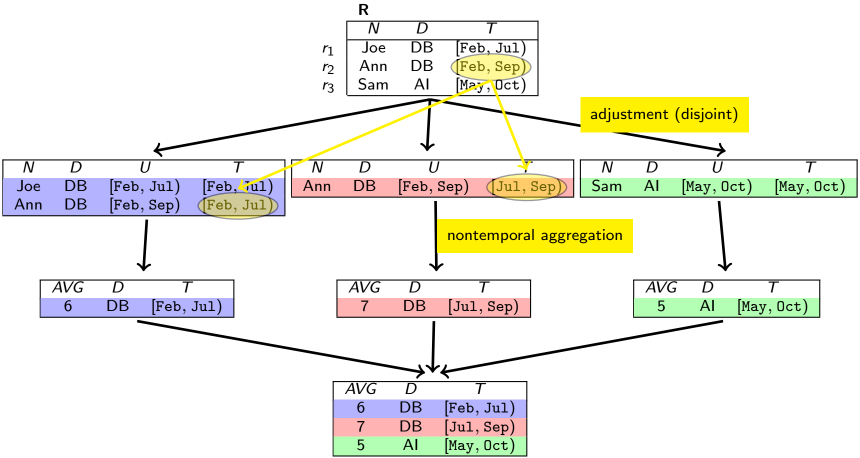
AVG	D	T
5	AI	[May, Oct)

AVG	D	T
6	DB	[Feb, Jul)
7	DB	[Jul, Sep)
5	AI	[May, Oct)

- One input tuple contributes to at most one result tuple per month.

Temporal Splitter

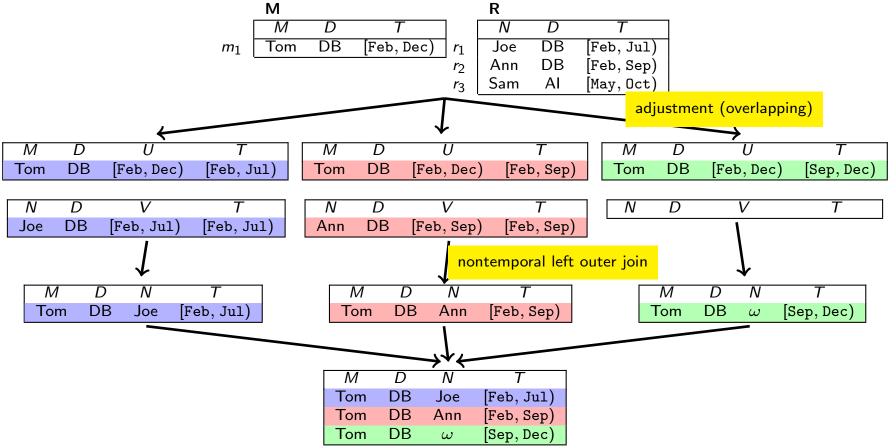
- Average duration of contracts per department: ${}_D\vartheta^T_{AVG(DUR(T))}(\mathbf{R})$



- One input tuple contributes to at most one result tuple per month.

Temporal Aligner

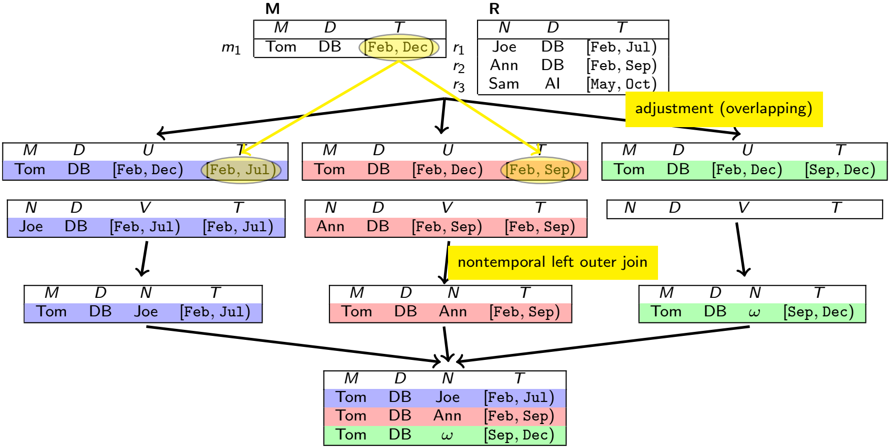
- Employees managed by manager: $\mathbf{M} \,⟕^{T}_{\mathbf{M}.D=\mathbf{R}.D}\, \mathbf{R}$



- One input tuple contributes to more than one result tuple per month. E.g., m_1 contributes twice to month Feb.

Temporal Aligner

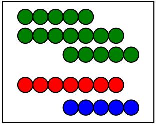
- Employees managed by manager: $\mathbf{M} \bowtie^{T}_{\mathbf{M}.D=\mathbf{R}.D} \mathbf{R}$



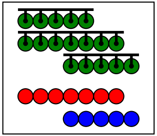
- One input tuple contributes to more than one result tuple per month. E.g., m_1 contributes twice to month Feb.

Properties of the Sequenced Semantics

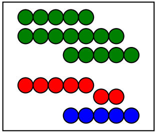
- Sequenced semantics for processing temporal data is defined over three properties:
  - A temporal query must be reducible to a nontemporal query (**Snapshot reducibility**)



  - Original Timestamps have to be accessible (**Extended snapshot reducibility**)



  - The boundaries of timestamps have to be preserved (**Change preservation**)



Snapshot Reducibility

- Constrains the result of a temporal operator ψ^T to the result of its corresponding nontemporal operator ψ applied at every snapshot.

- $\forall t : \tau_t(\psi^T(\mathbf{D^T})) \equiv \psi(\tau_t(\mathbf{D^T}))$
  - τ_t ...timeslice at t
  - $\mathbf{D^T}$ ...temporal database

- Ex: Time-varying Count corresponds to Count at each month.

R

	N	*D*	*T*
r_1	Joe	DB	[Feb, Jul)
r_2	Ann	DB	[Feb, Sep)
r_3	Sam	AI	[May, Oct)

$\tau_{\text{Feb}}(\mathbf{R})$

N	*D*
Joe	DB
Ann	DB

$_D\vartheta_{Count(*)}(\tau_{\text{Feb}}(\mathbf{R}))$

Count	*D*
2	DB

...

$\tau_{\text{Jul}}(\mathbf{R})$

N	*D*
Ann	DB
Sam	AI

$_D\vartheta_{Count(*)}(\tau_{\text{Jul}}(\mathbf{R}))$

Count	*D*
1	DB
1	AI

...

Count	*D*	*T*
2	DB	[Feb, Jul)
1	DB	[Jul, Sep)
1	AI	[May, Oct)

Extended Snapshot Reducibility

- Constrains the result of a temporal operator ψ^T to the result of its corresponding nontemporal operator ψ applied at every snapshot with original timestamps.

- $\forall t : \tau_t(\psi^T(\mathbf{D^T})) \equiv \psi(\tau_t(\epsilon(\mathbf{D^T})))$
  - τ_t ... timeslice at t
  - $\mathbf{D^T}$ ... temporal database
  - ϵ ... propagate original timestamp

R

	N	D	T
r_1	Joe	DB	[Feb, Jul)
r_2	Ann	DB	[Feb, Sep)
r_3	Sam	AI	[May, Oct)

$\tau_{\texttt{Feb}}(\epsilon_U(\mathbf{R}))$

U	N	D
[Feb, Jul)	Joe	DB
[Feb, Sep)	Ann	DB

$_D\vartheta_{AVG(DUR(U))}(\tau_{\texttt{Feb}}(\epsilon_U(\mathbf{R})))$

AVG	D
6	DB

...

$\tau_{\texttt{Jul}}(\epsilon_U(\mathbf{R}))$

U	N	D
[Feb, Sep)	Ann	DB
[May, Oct)	Sam	AI

$_D\vartheta_{AVG(DUR(U))}(\tau_{\texttt{Jul}}(\epsilon_U(\mathbf{R})))$

AVG	D
7	DB
5	AI

AVG	D	T
6	DB	[Feb, Jul)
7	DB	[Jul, Sep)
5	AI	[May, Oct)

...

Change Preservation/1

- Constrains the timestamps in the result of a temporal operator ψ^T by its lineage information.
  1. Lineage set $\mathrm{L}[\psi^T(\mathbf{D^T})](z, t)$ over all time points $z.T$ is equal.
  2. Maximal timestamps w.r.t **1**.

R

	N	D	T
r_1	Joe	DB	[Feb, Jul)
r_2	Ann	DB	[Feb, Sep)
r_3	Sam	AI	[May, Oct)

AVG	D	T
6	DB	[Feb, Jul)
7	DB	[Jul, Sep)
5	AI	[May, Oct)

- Change preservation defines the timestamps in the result.

Change Preservation/2

- $\mathrm{L}[{}_D\vartheta^T_{AVG(DUR(T))}(\mathbf{R})](z,t) = \langle\{r \in \mathbf{R} \mid z.D = r.D \wedge t \in r.T\}\rangle$

R

	N	D	T
r_1	Joe	DB	[Feb, Jul)
r_2	Ann	DB	[Feb, Sep)
r_3	Sam	AI	[May, Oct)

	AVG	D	T
z_1	6	DB	[Feb, Jul)
z_2	7	DB	[Jul, Sep)
z_3	5	AI	[May, Oct)

- $z_1 = (6, \mathrm{DB}, [\mathtt{Feb}, \mathtt{Jul}))$
  - $\mathrm{L}[{}_D\vartheta^T_{AVG(DUR(T))}(\mathbf{R})](z_1, \mathtt{Feb}) = \langle\{r_1, r_2\}\rangle$
  - $\mathrm{L}[{}_D\vartheta^T_{AVG(DUR(T))}(\mathbf{R})](z_1, \mathtt{Mar}) = \langle\{r_1, r_2\}\rangle$
  - $\mathrm{L}[{}_D\vartheta^T_{AVG(DUR(T))}(\mathbf{R})](z_1, \mathtt{Apr}) = \langle\{r_1, r_2\}\rangle$
  - $\mathrm{L}[{}_D\vartheta^T_{AVG(DUR(T))}(\mathbf{R})](z_1, \mathtt{May}) = \langle\{r_1, r_2\}\rangle$
  - $\mathrm{L}[{}_D\vartheta^T_{AVG(DUR(T))}(\mathbf{R})](z_1, \mathtt{Jun}) = \langle\{r_1, r_2\}\rangle$
- $z_2 = (7, \mathrm{DB}, [\mathtt{Jul}, \mathtt{Sep}))$
  - $\mathrm{L}[{}_D\vartheta^T_{AVG(DUR(T))}(\mathbf{R})](z_2, \mathtt{Jul}) = \langle\{r_2\}\rangle$
  - $\mathrm{L}[{}_D\vartheta^T_{AVG(DUR(T))}(\mathbf{R})](z_2, \mathtt{Aug}) = \langle\{r_2\}\rangle$

Change Preservation/3

- Change preservation allows to have values that are only valid for the entire interval.
- When adjusting timestamps such values can be scaled.

- Project budgets B of departments D during time T.

P

	B	P	D	T
p_1	10k	P1	DB	[Feb, Jul)
p_2	21k	P2	DB	[Feb, Sep)
p_3	15k	P3	AI	[May, Oct)

P

B	P	D	T
10k	P1	DB	[Feb, Jul)
15k	P2	DB	[Feb, Jul)
6k	P2	DB	[Jul, Sep)
15k	P3	AI	[May, Oct)

Reduction Rules

Reduction: $\psi^T \rightarrow (\mathcal{N}|\phi) \rightarrow \psi$

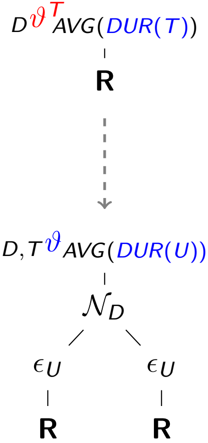
Operator			Reduction
Selection	$\sigma^T_\theta(\mathbf{r})$	$=$	$\sigma_\theta(\mathbf{r})$
Projection	$\pi^T_{\mathbf{B}}(\mathbf{r})$	$=$	$\pi_{\mathbf{B},T}(\mathcal{N}_{\mathbf{B}}(\mathbf{r},\mathbf{r}))$
Aggregation	${}_{\mathbf{B}}\vartheta^T_F(\mathbf{r})$	$=$	${}_{\mathbf{B},T}\vartheta_F(\mathcal{N}_{\mathbf{B}}(\mathbf{r},\mathbf{r}))$
Difference	$\mathbf{r} -^T \mathbf{s}$	$=$	$\mathcal{N}_{\mathbf{A}}(\mathbf{r},\mathbf{s}) - \mathcal{N}_{\mathbf{A}}(\mathbf{s},\mathbf{r})$
Union	$\mathbf{r} \cup^T \mathbf{s}$	$=$	$\mathcal{N}_{\mathbf{A}}(\mathbf{r},\mathbf{s}) \cup \mathcal{N}_{\mathbf{A}}(\mathbf{s},\mathbf{r})$
Intersection	$\mathbf{r} \cap^T \mathbf{s}$	$=$	$\mathcal{N}_{\mathbf{A}}(\mathbf{r},\mathbf{s}) \cap \mathcal{N}_{\mathbf{A}}(\mathbf{s},\mathbf{r})$
Cart. Prod.	$\mathbf{r} \times^T \mathbf{s}$	$=$	$\alpha(\phi_\top(\mathbf{r},\mathbf{s}) \bowtie_{\mathbf{r}.T=\mathbf{s}.T} \phi_\top(\mathbf{s},\mathbf{r}))$
Inner Join	$\mathbf{r} \bowtie^T_\theta \mathbf{s}$	$=$	$\alpha(\phi_\theta(\mathbf{r},\mathbf{s}) \bowtie_{\theta\wedge\mathbf{r}.T=\mathbf{s}.T} \phi_\theta(\mathbf{s},\mathbf{r}))$
Left O. Join	$\mathbf{r}$ ⟕$^T_\theta$ $\mathbf{s}$	$=$	$\alpha(\phi_\theta(\mathbf{r},\mathbf{s})$ ⟕$_{\theta\wedge\mathbf{r}.T=\mathbf{s}.T}$ $\phi_\theta(\mathbf{s},\mathbf{r}))$
Right O. Join	$\mathbf{r}$ ⟖$^T_\theta$ $\mathbf{s}$	$=$	$\alpha(\phi_\theta(\mathbf{r},\mathbf{s})$ ⟖$_{\theta\wedge\mathbf{r}.T=\mathbf{s}.T}$ $\phi_\theta(\mathbf{s},\mathbf{r}))$
Full O. Join	$\mathbf{r}$ ⟗$^T_\theta$ $\mathbf{s}$	$=$	$\alpha(\phi_\theta(\mathbf{r},\mathbf{s})$ ⟗$_{\theta\wedge\mathbf{r}.T=\mathbf{s}.T}$ $\phi_\theta(\mathbf{s},\mathbf{r}))$
Anti Join	$\mathbf{r} \rhd^T_\theta \mathbf{s}$	$=$	$\phi_\theta(\mathbf{r},\mathbf{s}) \rhd_{\theta\wedge\mathbf{r}.T=\mathbf{s}.T} \phi_\theta(\mathbf{s},\mathbf{r})$

α ... temporal duplicate elimination.

Constructing Sequenced Algebra Expressions

Query: ${}_{D}\vartheta^{T}_{AVG(DUR(T))}(\mathbf{R})$

1. Timestamp propagation:
   ${}_{D}\vartheta^{T}_{AVG(DUR(T))}(\epsilon_U(\mathbf{R}))$
2. Timestamp substitution:
   ${}_{D}\vartheta^{T}_{AVG(DUR(U))}(\epsilon_U(\mathbf{R}))$
3. Temporal adjustment:
   $\mathbf{R}' \leftarrow \mathcal{N}_D(\epsilon_U(\mathbf{R}), \epsilon_U(\mathbf{R}))$
4. Nontemporal aggregation:
   ${}_{D,T}\vartheta_{AVG(DUR(U))}(\mathbf{R}')$



Constructing Sequenced Algebra Expressions

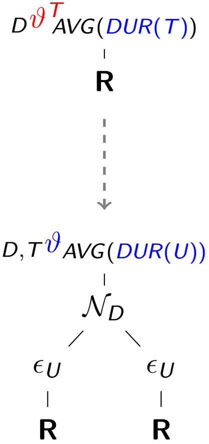
Query: ${}_{D}\vartheta^{T}_{AVG(DUR(T))}(\mathbf{R})$

1. Timestamp propagation:
   ${}_{D}\vartheta^{T}_{AVG(DUR(T))}(\epsilon_U(\mathbf{R}))$

2. Timestamp substitution:
   ${}_{D}\vartheta^{T}_{AVG(DUR(U))}(\epsilon_U(\mathbf{R}))$

3. Temporal adjustment:
   $\mathbf{R}' \leftarrow \mathcal{N}_D(\epsilon_U(\mathbf{R}), \epsilon_U(\mathbf{R}))$

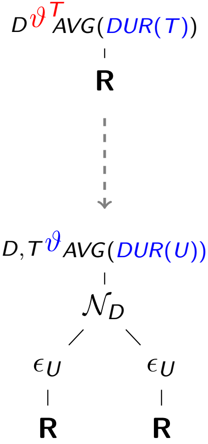
4. Nontemporal aggregation:
   ${}_{D,T}\vartheta_{AVG(DUR(U))}(\mathbf{R}')$



Constructing Sequenced Algebra Expressions

Query: ${}_{D}\vartheta^{T}_{AVG(DUR(T))}(\mathbf{R})$

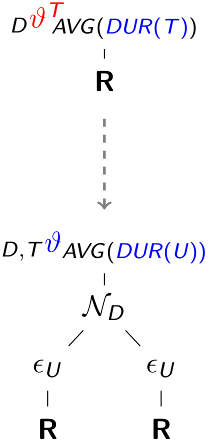
1. Timestamp propagation:
   ${}_{D}\vartheta^{T}_{AVG(DUR(T))}(\epsilon_U(\mathbf{R}))$
2. Timestamp substitution:
   ${}_{D}\vartheta^{T}_{AVG(DUR(U))}(\epsilon_U(\mathbf{R}))$
3. Temporal adjustment:
   $\mathbf{R}' \leftarrow \mathcal{N}_D(\epsilon_U(\mathbf{R}), \epsilon_U(\mathbf{R}))$
4. Nontemporal aggregation:
   ${}_{D,T}\vartheta_{AVG(DUR(U))}(\mathbf{R}')$



Constructing Sequenced Algebra Expressions

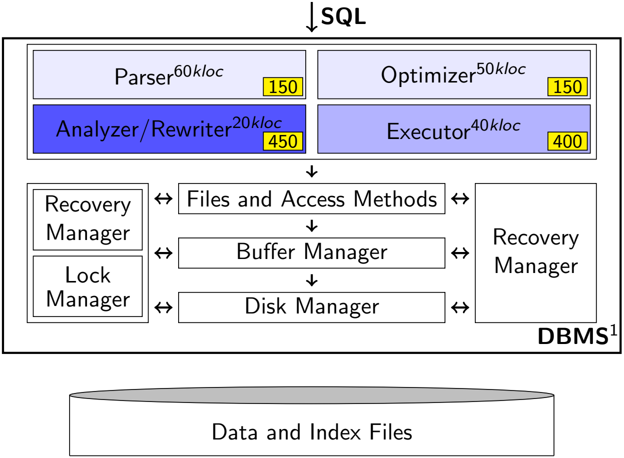
Query: ${}_{D}\vartheta^{T}_{AVG(DUR(T))}(\mathbf{R})$

1. Timestamp propagation:
   ${}_{D}\vartheta^{T}_{AVG(DUR(T))}(\epsilon_U(\mathbf{R}))$
2. Timestamp substitution:
   ${}_{D}\vartheta^{T}_{AVG(DUR(U))}(\epsilon_U(\mathbf{R}))$
3. Temporal adjustment:
   $\mathbf{R}' \leftarrow \mathcal{N}_D(\epsilon_U(\mathbf{R}), \epsilon_U(\mathbf{R}))$
4. Nontemporal aggregation:
   ${}_{D,T}\vartheta_{AVG(DUR(U))}(\mathbf{R}')$



PostgreSQL Implementation/1

- DBMS kernel integration of temporal primitives.



[1] Image: Raghu Ramakrishnan and Johannes Gehrke. *Database Management Systems.* McGraw-Hill 2003

PostgreSQL Implementation/2

- Our prototype provides a direct access to primitive operators:

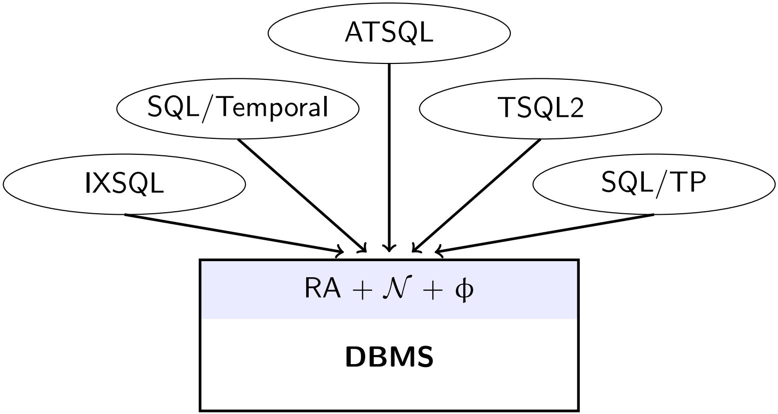
$$\begin{aligned}
\epsilon_U(\mathbf{r}) &: \texttt{SELECT Ts Us, Te Ue, * FROM r} \\
\mathcal{N}_{\mathbf{B}}(\mathbf{r}, \mathbf{s}) &: \texttt{FROM (r NORMALIZE s USING(}\mathbf{B}\texttt{)) r} \\
\phi_\theta(\mathbf{r}, \mathbf{s}) &: \texttt{FROM (r ALIGN s ON } \theta\texttt{) r} \\
\alpha(\mathbf{r}) &: \texttt{SELECT ABSORB * FROM r}
\end{aligned}$$

- Temporal SQL languages can be implemented in Parser/Analyzer.

Source Code: `http://www.ifi.uzh.ch/dbtg/research/align.html`

Algebraic Basis for Sequenced Semantics

- Reduction is at algebra level.
  $\Rightarrow$ Any existing language supporting sequenced semantics can be implemented.



Empirical Evaluation

- Datasets
  - Real world dataset Incumben University of Arizona
  - Synthetic datasets

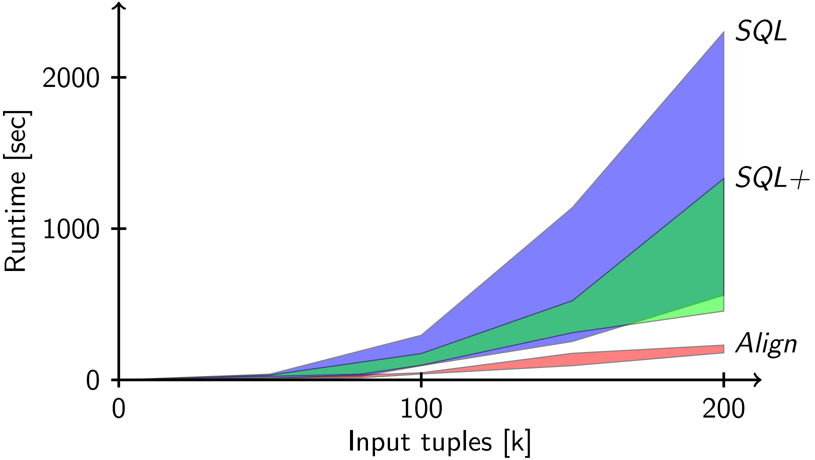
- Comparison on Outer Joins

Align	Temporal Alignment and Reduction Rules.
SQL	Plain SQL solution[2]
SQL+	SQL Join and Difference based on $\mathcal{N}$

[2] R. T. Snodgrass. *Developing Time-Oriented Database Applications in SQL.* Morgen Kaufmann Publisher, 1999.

Outer Joins

- Real world and random datasets.
- Equi-Full and Theta-Left Outer Joins.

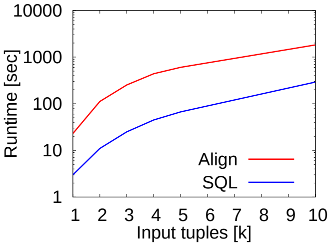


- SQL is inefficient and not robust for timestamp adjustment.

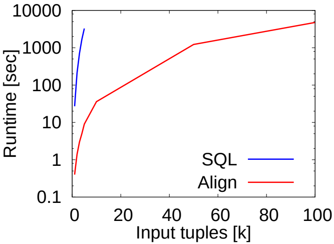
Outer Joins SQL

- Left Outer Join ($\theta = \texttt{true}$).

  - All timestamps equal.



  - All timestamps disjoint.



- *SQL* adjustment is based on `NOT EXISTS`.
- *SQL* efficient when all timestamps are equal.
  - Every tuple stops `NOT EXISTS`.
- *SQL* inefficient when all timestamps are disjoint.
  - All tuples need to be analyzed to stop `NOT EXISTS`.

Related Work

Nontemporal Semantics

- Timestamps are explicit.
- $_{D}\vartheta_{AVG(DUR(T))}(\mathbf{R})$
- State of the art in nontemporal databases.

Snapshot Semantics

- Timestamps are implicit.
- $_{D}\vartheta^{T}_{Count(*)}(\mathbf{R})$
- State of the art in temporal databases.

Sequenced Semantics

- Timestamps are explicit and implicit.
- $_{D}\vartheta^{T}_{AVG(DUR(T))}(\mathbf{R})$

Related Work: Nontemporal Semantics

- Standard SQL (SQL-2)
  `DATE` datatype with predicates $(=, <, >)$
- Temporal Postgres, Oracle Workspace Manager, Teradata 13.10
  `PERIOD` datatype, `INTERSECT`, `INTERVAL LENGTH` functions, ...

- Timestamps are explicit.
  - $_{D}\vartheta_{AVG(DUR(T))}(\mathbf{R})$
  - $\mathbf{R} ⟕_{Min \leq DUR(\mathbf{R}.T) \leq Max} \mathbf{P}$

- Achieving snapshot reducibility is difficult and inefficient.

Related Work: Snapshot Semantics

- N. A. Lorentzos and Y. G. Mitsopoulos. *SQL Extension for Interval Data*. IEEE Trans. Knowl. Data Eng. 1997.
- D. Toman. *Point-Based Temporal Extensions of SQL and Their Efficient Implementation*. Temporal Databases: Research and Practice Springer Verlag. 1998.
- W. Li, R. T. Snodgrass, S. Deng, V. K. Gattu, A. Kasthurirangan: *Efficient Sequenced Integrity Constraint Checking*. ICDE. 2001

- Timestamps are implicit.
  - ${}_{D}\vartheta^{T}_{Count(*)}(\mathbf{R})$
  - $\mathbf{R} \times^{T} \mathbf{R}$

- Original timestamps are not available in snapshots.
- Change preservation can not be represented.

Related Work: Sequenced Semantics

- M. H. Böhlen and C. S. Jensen and R. T. Snodgrass. *Temporal Statement Modifiers*. ACM TODS. 2000.

- Timestamps are explicit and implicit.
  - ${}_{D}\vartheta^{T}_{AVG(DUR(T))}(\mathbf{R})$
  - $\mathbf{R} \bowtie^{T}_{Min \leq DUR(\mathbf{R}.T) \leq Max} \mathbf{P}$

- Partial support for accessing original timestamps:

  Cartesian product that propagates original timestamps.

Summary and Future Work

Summary

- **Comprehensive algebraic basis** for the sequenced semantics, where timestamps are explicit and implicit: ${}_{D}\vartheta^{T}_{AVG(DUR(T))}(\mathbf{R})$
- Two algebraic primitives for **adjustment of timestamps**: $\mathcal{N}$, ϕ
- **Reduction rules** from temporal RA to nontemporal RA
- **Timestamp propagation** for accessing original timestamps
- Deep integration into DBMS kernel of PostgreSQL.

Future Work

- Optimization/equivalence rules for temporal primitives
- Extensions towards time depended (malleable) quantities

Summary and Future Work

Summary

- **Comprehensive algebraic basis** for the sequenced semantics, where timestamps are explicit and implicit: ${}_{D}\vartheta^{T}_{AVG(DUR(T))}(\mathbf{R})$
- Two algebraic primitives for **adjustment of timestamps**: $\mathcal{N}$, ϕ
- **Reduction rules** from temporal RA to nontemporal RA
- **Timestamp propagation** for accessing original timestamps
- Deep integration into DBMS kernel of PostgreSQL.

Future Work

- Optimization/equivalence rules for temporal primitives
- Extensions towards time depended (malleable) quantities

Thank You!