

Temporal and Spatial Data Management

Fall 2019

Sequenced Semantics

SL03

- ▶ Elements of the sequenced semantics
- ▶ Integration of the sequenced semantics into the DBMS kernel
- ▶ SQL implementation

Table of Contents

- ▶ Elements of the sequenced semantics
 - 1) ▶ Snapshot reducibility
 - 2) ▶ Change preservation
 - 3) ▶ Extended snapshot reducibility
- ▶ Integration of the sequenced semantics into the DBMS kernel
 - ▶ Algebraic operators
 - ▶ Reduction rules
- ▶ SQL implementation

Literature

- ▶ A. Dignös, M. H. Böhlen, J. Gamper, C. S. Jensen, Extending the Kernel of a Relational DBMS with Comprehensive Support for Sequenced Temporal Queries, ACM Transactions on Database Systems 41(4): 26:1-26:46, 2016.

Idea and Goal of Sequenced Semantics

- ▶ There is a close relationship between a temporal and a nontemporal database:
 - ▶ the snapshot of a temporal relation at a time t is a nontemporal relation.
- ▶ All nontemporal statements can be evaluated at each snapshot of a temporal database ("at each time point")
- ▶ There should be a close relationship between a temporal and a nontemporal statement:
 - ▶ a temporal aggregation should resemble a nontemporal aggregation.
- ▶ With SQL this is not the case (remember temporal join versus join).

Setup and Notations

- ▶ Relation schema: $R(A_1, \dots, A_n, T_S, T_E)$
 - ▶ $A_1, \dots, A_n$ are the explicit attributes
 - ▶ T_S, T_E are temporal attributes
 - ▶ T_S : valid time start
 - ▶ T_E : valid time end
 - ▶ z^{n+2} denotes a tuple of arity $n + 2$
 - ▶ We assume half open intervals: $[T_S, T_E)$
 - ▶ We write T to refer to the period $[T_S, T_E)$
 - ▶ $t \in T \equiv T_S \leq t < T_E$
- 

The Timeslice Operator

2019/7/23

- ▶ The timeslice operator maps a temporal to a nontemporal relation.
- ▶ Definition of the **timeslice operator**:

$$\tau_t(r) = \{ \underline{z^{(n)}} \mid \exists x \in r (z.A = x.A \wedge x.T_S \leq t < x.T_E) \}$$

- ▶ Example: checkout1

CustID	TapeNum	T
C101	T3456	[19,20]
C102	T1245	[20,22]

- ▶ Apply the timeslice operator at various points:

...

$$\tau_{18}(\text{checkout1}) = \{\}$$

$$\tau_{19}(\text{checkout1}) = \{(C101, T3456)\}$$

$$\tau_{20}(\text{checkout1}) = \{(C101, T3456), (C102, T1245)\}$$

$$\tau_{21}(\text{checkout1}) = \{(C102, T1245)\}$$

CustID	TapeNum
C101	T3456

CustID	TapeNum
C101	T3456
C102	T1245

Snapshot Equivalence

- ▶ Two temporal relations, r and s , are snapshot equivalent, $r \stackrel{s}{\equiv} s$, iff for all times t their snapshots are identical.
- ▶ Definition of **snapshot equivalence**: $r \stackrel{s}{\equiv} s$ iff $\forall t(\tau_t(r) = \tau_t(s))$

Snapshot Equivalence

- ▶ Two temporal relations, r and s , are snapshot equivalent, $r \stackrel{s}{\equiv} s$, iff for all times t their snapshots are identical.
- ▶ Definition of **snapshot equivalence**: $r \stackrel{s}{\equiv} s$ iff $\forall t(\tau_t(r) = \tau_t(s))$
- ▶ **Example**: Two-day and four-day checkouts in the video example:

CustID	TapeNum	T
C102	T1245	[19,20]
C102	T1245	[21,22]

CustID	TapeNum	T
C102	T1245	[19,22]

- ▶ Apply the timeslice operator at each time point:

$$\tau_{18}(\text{checkout1}) = \{\}$$

$$\tau_{19}(\text{checkout1}) = \{(C102, T1245)\}$$

$$\tau_{20}(\text{checkout1}) = \{(C102, T1245)\}$$

...

$$\tau_{18}(\text{checkout2}) = \{\}$$

$$\tau_{19}(\text{checkout2}) = \{(C102, T1245)\}$$

$$\tau_{20}(\text{checkout2}) = \{(C102, T1245)\}$$

...

Snapshot Equivalence

- ▶ Two temporal relations, r and s , are snapshot equivalent, $r \stackrel{s}{\equiv} s$, iff for all times t their snapshots are identical.
- ▶ Definition of **snapshot equivalence**: $r \stackrel{s}{\equiv} s$ iff $\forall t(\tau_t(r) = \tau_t(s))$
- ▶ **Example**: Two-day and four-day checkouts in the video example:

checkout1				checkout2		
CustID	TapeNum	T		CustID	TapeNum	T
C102	T1245	[19,20]		C102	T1245	[19,22]
C102	T1245	[21,22]	≠			

- ▶ Apply the timeslice operator at each time point:

$$\tau_{18}(\text{checkout1}) = \{\}$$

$$\tau_{18}(\text{checkout2}) = \{\}$$

$$\tau_{19}(\text{checkout1}) = \{(C102, T1245)\}$$

$$\tau_{19}(\text{checkout2}) = \{(C102, T1245)\}$$

$$\tau_{20}(\text{checkout1}) = \{(C102, T1245)\}$$

$$\tau_{20}(\text{checkout2}) = \{(C102, T1245)\}$$

...

...

- ▶ checkout1 and checkout2 are snapshot equivalent. 
- ▶ checkout1 and checkout2 are syntactically different. 

Snapshot Reducibility (1)

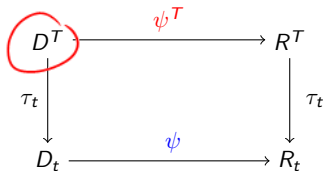
Snapshot reducibility reduces the semantics of **temporal operators** to the semantics of the corresponding **nontemporal operators**.

ψ^T is **snapshot reducible** to ψ iff for all t :

temporal count $\tau_t(\psi^T(R_1, \dots, R_n)) \equiv \psi(\tau_t(R_1), \dots, \tau_t(R_n))$ *nontemporal count*

Illustration of snapshot reducibility:

► $\forall t : \tau_t(\psi^T(D^T)) = \psi(\tau_t(D^T))$



- D^T = temporal DB
- $\psi^T \in \{\sigma^T, \pi^T, \vartheta^T, \times^T, \cup^T, -^T\}$
- R^T = temporal result relation
- τ_t = snapshot at time point t
- D_t = snapshot of D^T at time t
- $\psi \in \{\sigma, \pi, \vartheta, \times, \cup, -\}$
- R_t = result relation at time t

Reducibility of Temporal Operators

- ▶ For each **non-temporal operator** there is a **snapshot reducible operator**:
 - ▶ count at each point in time
 - ▶ join at each point in time
 - ▶ primary key at each point in time
 - ▶ ranking at each point in time
- ▶ We write ψ^T for the snapshot counterpart of the relational algebra operator ψ .
- ▶ Ex: A **temporal count** ϑ^T corresponds to a **nontemporal count** ϑ at each point in time.

Reducibility of Temporal Operators

$\{2,3,4,5,6, 12,13,14\} \Rightarrow 2-6, 12-14$

- For each **non-temporal operator** there is a **snapshot reducible operator**:

- count at each point in time
- join at each point in time
- primary key at each point in time
- ranking at each point in time

- across time (do not split up)

- how to form periods

- pts and predicates on time
T before [5, 12]

- We write ψ^T for the snapshot counterpart of the relational algebra operator ψ .
- Ex: A **temporal count** ϑ^T corresponds to a **nontemporal count** ϑ at each point in time.
- **Snapshot reducibility** ensures a systematic generalization of all nontemporal operators to temporal operators.

- $D\vartheta_{SUM} \dots \Rightarrow D\vartheta^T_{SUM} \dots$
- $\bowtie_{\theta} \dots \Rightarrow \bowtie^T_{\theta} \dots$

- 4

Data Lineage/1

SR defines result tuples at each line point
(or which line points are in the result)

The **lineage set**, $L[\psi^T(R_1, \dots, R_n)](z, t)$, of result tuple z at time t is the set of witness lists of argument tuples, $\{\langle r_1, \dots, r_n \rangle\}$, $r_i \in R_i$, from which z is derived:

Lineage defines how to group points into periods

$$L[\sigma_\theta^T(R)](z, t) = \{\langle r \rangle \mid r \in R \wedge z.A = r.A \wedge t \in r.T\}$$

$$L[\pi_B^T(R)](z, t) = \{\langle r \rangle \mid r \in R \wedge z.B = r.B \wedge t \in r.T\}$$

$$L[B \vartheta_F^T(R)](z, t) = \{\langle r \rangle \mid r \in R \wedge z.B = r.B \wedge t \in r.T\}$$

$$L[R -^T S](z, t) = \{\langle r, \perp \rangle \mid r \in R \wedge z.A = r.A \wedge t \in r.T\}$$

$$L[R \cup^T S](z, t) = \{\langle r, \perp \rangle \mid r \in R \wedge z.A = r.A \wedge t \in r.T\} \cup \\ \{\langle \perp, s \rangle \mid s \in S \wedge z.A = s.A \wedge t \in s.T\}$$

$$L[R \times^T S](z, t) = \{\langle r, s \rangle \mid r \in R \wedge z.A = r.A \wedge t \in r.T \wedge \\ s \in S \wedge z.C = s.C \wedge t \in s.T\}$$

$$L[R \triangleright_\theta^T S](z, t) = \{\langle r, \perp \rangle \mid r \in R \wedge z.A = r.A \wedge t \in r.T\}$$

Consider query $Q = D \overset{T}{\vee}_{CNT(*)}(R)$:

	<u>R</u>			
	N	D	B	T
r_1	P_1	CS	5K	[1, 6)
r_2	P_2	CS	6K	[4, 7)
r_3	P_3	MA	2K	[1, 3)

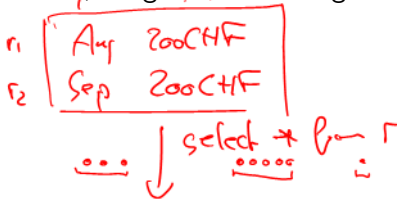
Result

	D	CNT	T
z_1	CS	1	[1, 4)
z_2	CS	2	[4, 6)
z_3	CS	1	[6, 7)
z_4	MA	1	[1, 3)

$CS \ 1 \ [1, 2)$
 $CS \ 1 \ [2, 4)$
 $\vdots$

For z_1 , z_2 , and z_3 and time points 3, 4, 5 and 6, we get the following lineage sets:

- ▶ $L[D \overset{T}{\vee}_{CNT(*)}(R)](z_1, 3) = \{\langle r_1 \rangle\}$
- ▶ $L[D \overset{T}{\vee}_{CNT(*)}(R)](z_2, 4) = \{\langle r_1 \rangle, \langle r_2 \rangle\}$
- ▶ $L[D \overset{T}{\vee}_{CNT(*)}(R)](z_2, 5) = \{\langle r_1 \rangle, \langle r_2 \rangle\}$
- ▶ $L[D \overset{T}{\vee}_{CNT(*)}(R)](z_3, 6) = \{\langle r_2 \rangle\}$



We use lineage to preserve changes (start and end of intervals).

- ▶ $\{(DB, 5K, [Feb, Jul])\} \neq \{(DB, 5K, [Feb, Apr]), (DB, 5K, [Apr, Jul])\}$
- ▶ makes a duration function natural

ψ^T is **change preserving** iff $\forall z, z' \in \psi^T(R_1, \dots, R_n)$:

$$\begin{aligned} & \forall t, t' \in z.T (L(z, t) = L(z, t')) \wedge \\ & (z.T_S - 1 \in z'.T \Rightarrow L(z', z.T_S - 1) \neq L(z, z.T_S)) \wedge \\ & (z.T_E \in z'.T \Rightarrow L(z', z.T_E) \neq L(z, z.T_S)) \end{aligned}$$

Change Preservation (2)

We use lineage to preserve changes (start and end of intervals).

- ▶ $\{(DB, 5K, [Feb, Jul])\} \neq \{(DB, 5K, [Feb, Apr]), (DB, 5K, [Apr, Jul])\}$
- ▶ makes a duration function natural

ψ^T is **change preserving** iff $\forall z, z' \in \psi^T(R_1, \dots, R_n)$:

$$\begin{aligned} & \forall t, t' \in z.T (L(z, t) = L(z, t')) \wedge \\ & (z.T_S - 1 \in z'.T \Rightarrow L(z', z.T_S - 1) \neq L(z, z.T_S)) \wedge \\ & (z.T_E \in z'.T \Rightarrow L(z', z.T_E) \neq L(z, z.T_S)) \end{aligned}$$

Change preservation permits facts that hold for an entire interval but not a subinterval. Intervals are not coalesced automatically.

- ▶ π_D^T

for two result tuples z, z'

only points with some lineage can be combined

before z lineage must be different

after z lineage must be diff

Extended Snapshot Reducibility

We want to access **time intervals** in **snapshot reducible operators**.

- at each time point join long and short projects

ψ^T is **extended snapshot reducible** iff for all t :

$$\tau_t(\psi^T(R_1, \dots, R_n)) \equiv \pi_{\mathbf{E}}(\psi(\tau_t(\epsilon_{U_1}(R_1)), \dots, \tau_t(\epsilon_{U_n}(R_n)))).$$

For $\psi \in \{\vartheta, \sigma, \pi, \times, \bowtie, \ltimes, \rtimes, \triangleright, \text{~~...~~}\}$, $\mathbf{E}$ = schema of $\psi^T(R_1, \dots, R_n)$.

make a copy
of T and
call it U_1

make a
copy of T
and call it
 U_n

Extended Snapshot Reducibility

We want to access **time intervals** in **snapshot reducible operators**.

- ▶ at each time point join long and short projects

ψ^T is **extended snapshot reducible** iff for all t :

$$\tau_t(\psi^T(R_1, \dots, R_n)) \equiv \pi_{\mathbf{E}}(\psi(\tau_t(\epsilon_{U_1}(R_1)), \dots, \tau_t(\epsilon_{U_n}(R_n)))).$$

For $\psi \in \{\vartheta, \sigma, \pi, \times, \bowtie, \ltimes, \rtimes, \triangleright, \triangleleft\}$, $\mathbf{E}$ = schema of $\psi^T(R_1, \dots, R_n)$.

Extended snapshot reducibility supports snapshot reducibility **and** access to timestamps (by propagating timestamps with the extend operator ϵ).

- ▶ at each time point the average duration of externally funded projects: $D^{\vartheta^T}_{AVG}(\text{DUR}(T))(R)$

old histogram

Table of Contents

► Elements of the sequenced semantics

- Snapshot reducibility 1)
- Change preservation 2)
- Extended snapshot reducibility 3)

easy, everybody agrees

► Integration of temporal support in DBMS kernel

- Algebraic adjustment operators
- Reduction rules

— difficult, little consensus

► SQL implementation

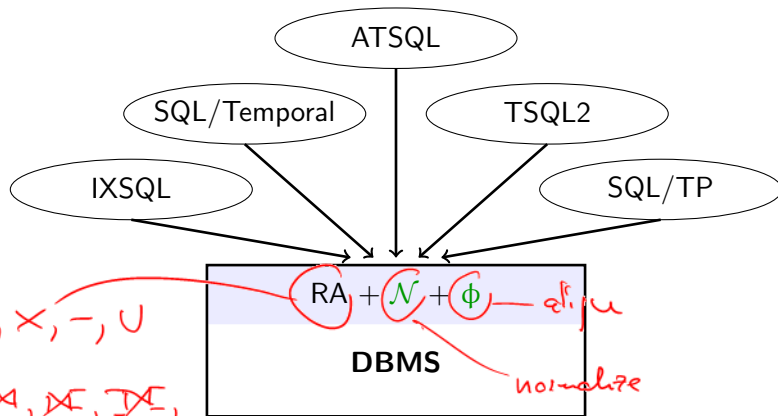
challenge: # of snapshots [

Integration of the Sequenced Semantics into the DBMS Kernel

- ▶ Goal:
 - ▶ Determine and implement in the DBMS kernel the functionality that is required to offer support for the sequenced semantics (snapshot reducibility + change preservation + extended snapshot reducibility).
 - ▶ The sequenced semantics supports all features that have been identified as important for processing time periods.
- ▶ Solution:
 - ▶ New **algebraic adjustment operators** that split periods into pieces, such that snapshot reducible queries only need equality for comparison.
 - ▶ Possibility to **propagate and access the original timestamps.**

Algebraic Basis for Sequenced Semantics

- Solution is at the level of the algebra.
⇒ Any language with a form of sequenced semantics is supported.



Temporal Query Processing

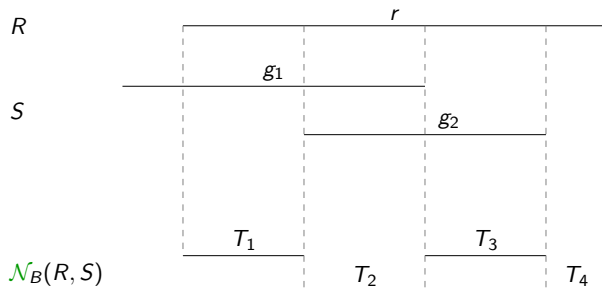
- ▶ To implement the sequenced semantics **two new algebra operators** for the **adjustment** of periods are needed:
 - ▶ Temporal normalization $\mathcal{N}$
 - ▶ Temporal alignment ϕ
- ▶ Adjustment respects the lineage.
- ▶ **Reduction rules** from temporal RA to nontemporal RA.
 - ▶ “How to use the adjustment operators”
- ▶ **Period propagation** ϵ makes it possible to access the original periods.
 - ▶ Used for extended snapshot reducibility

Temporal Adjustment

- ▶ The purpose of a temporal adjustment operators is to break periods into pieces.
- ▶ Two temporal adjustment operators are required since there are two classes of operators in relational DBMSs:
 - ▶ One input tuple contributes to **at most one** result tuple per time point.
⇒ **temporal normalization**
Example: Aggregation
 - ▶ One input tuple contributes to **more than one** result tuple per time point.
⇒ **temporal alignment**
Example: Joins

Temporal Normalization/1

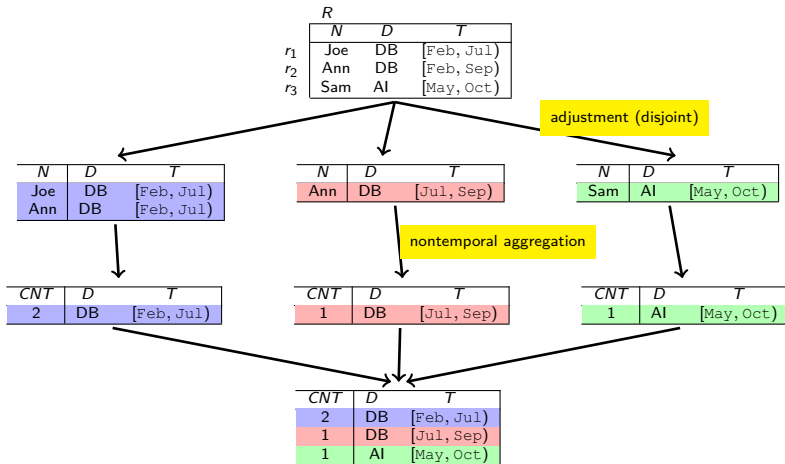
- **Normalize** splits each tuple in R with respect to the tuples in S that match on the grouping attributes B .



- Algebra: $\mathcal{N}_B(R, S)$
- R is split with respect to all relevant tuples in S .

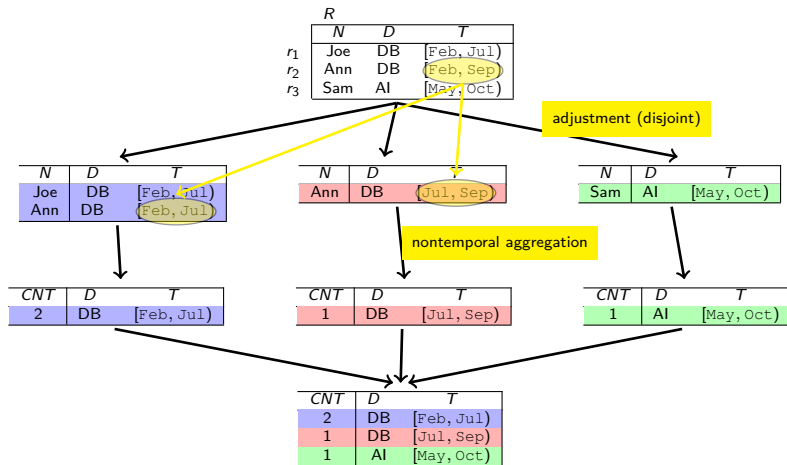
Temporal Normalization/2

- Number of contracts per department: $D \vartheta^T_{AVG}(DUR(T))(R)$



Temporal Normalization/2

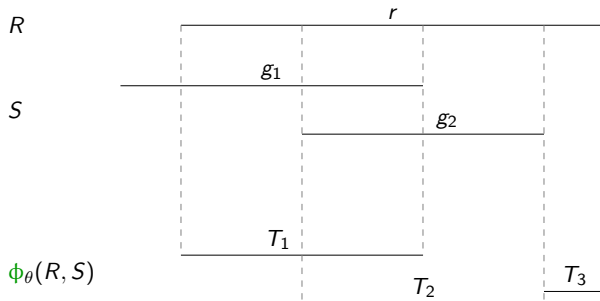
- Number of contracts per department: $D \vartheta^T_{AVG}(DUR(T))(R)$



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

Temporal Alignment/1

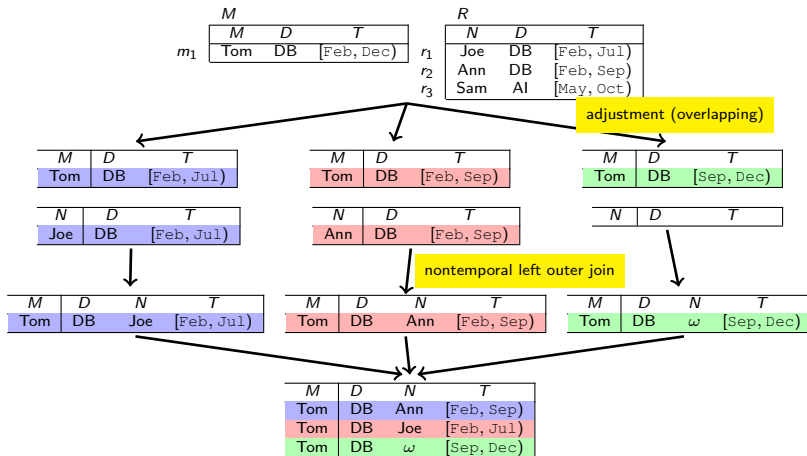
- **Align** splits each tuple in R with respect to each tuple in the group of tuples in S that satisfy θ .



- Algebra: $\phi_\theta(R, S)$
- R is split with respect to each relevant tuple in S .

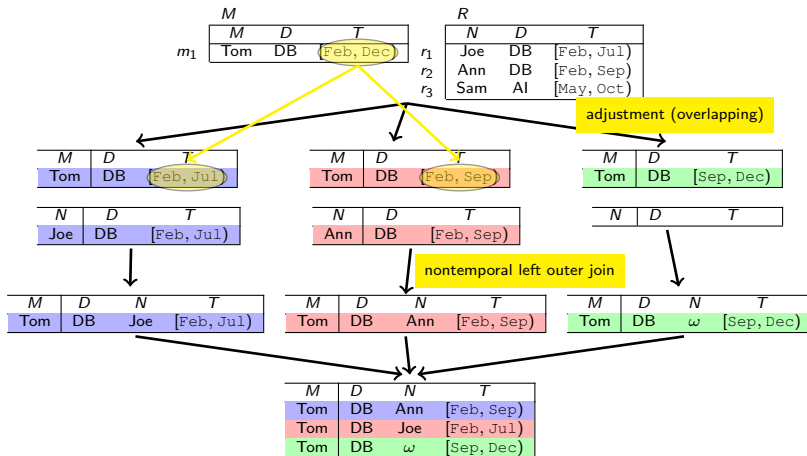
Temporal Alignment/2

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



Temporal Alignment/2

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



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

Reduction Rules

Operator	Reduction	
Selection	$\sigma_{\theta}^T(R)$	$= \sigma_{\theta}(R)$
Projection	$\pi_{\mathbf{B}}^T(R)$	$= \pi_{\mathbf{B},T}(\mathcal{N}_{\mathbf{B}}(R, R))$
Aggregation	$\mathbf{B} \vartheta_F^T(R)$	$= \mathbf{B},T \vartheta_F(\mathcal{N}_{\mathbf{B}}(R, R))$
Difference	$R -^T S$	$= \mathcal{N}_{\mathbf{A}}(R, S) - \mathcal{N}_{\mathbf{A}}(S, R)$
Union	$R \cup^T S$	$= \mathcal{N}_{\mathbf{A}}(R, S) \cup \mathcal{N}_{\mathbf{A}}(S, R)$
Intersection	$R \cap^T S$	$= \mathcal{N}_{\mathbf{A}}(R, S) \cap \mathcal{N}_{\mathbf{A}}(S, R)$
Cart. Prod.	$R \times^T S$	$= \phi_{\top}(R, S) \bowtie_{R.T=S.T} \phi_{\top}(S, R)$
Inner Join	$R \Join_{\theta}^T S$	$= \phi_{\theta}(R, S) \bowtie_{\theta \wedge R.T=S.T} \phi_{\theta}(S, R)$
Left O. Join	$R \Join_{\theta}^T S$	$= \phi_{\theta}(R, S) \Join_{\theta \wedge R.T=S.T} \phi_{\theta}(S, R)$
Right O. Join	$R \Join_{\theta}^T S$	$= \phi_{\theta}(R, S) \Join_{\theta \wedge R.T=S.T} \phi_{\theta}(S, R)$
Full O. Join	$R \Join_{\theta}^T S$	$= \phi_{\theta}(R, S) \Join_{\theta \wedge R.T=S.T} \phi_{\theta}(S, R)$
Anti Join	$R \Join_{\theta}^T S$	$= \phi_{\theta}(R, S) \Join_{\theta \wedge R.T=S.T} \phi_{\theta}(S, R)$

Answering Sequenced Queries/1

Approach for formulating sequenced queries:

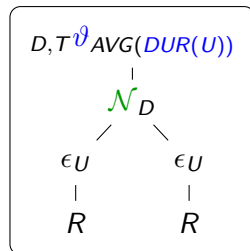
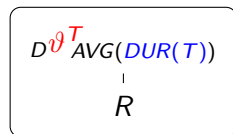
1. Formulate query without thinking about time/periods; make all operators temporal (e.g., ϑ^T).
2. Add copies of periods that are needed later (e.g., in selection conditions).
3. Replace all references to periods with references to these copies.
4. Apply reduction rules to get nontemporal relational algebra expression.

Answering Sequenced Queries/2

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

1. Timestamp propagation:

$$D \vartheta^T_{AVG}(DUR(T))(\epsilon_U(R))$$



Answering Sequenced Queries/2

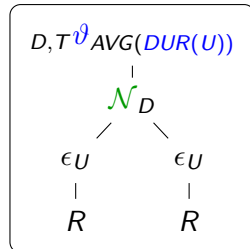
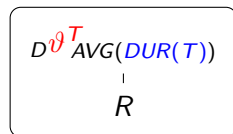
Query: $D^{\vartheta^T}_{AVG}(DUR(T))(R)$

1. Timestamp propagation:

$$D^{\vartheta^T}_{AVG}(DUR(T))(\epsilon_U(R))$$

2. Timestamp substitution:

$$D^{\vartheta^T}_{AVG}(DUR(U))(\epsilon_U(R))$$



Answering Sequenced Queries/2

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

1. Timestamp propagation:

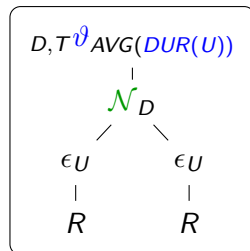
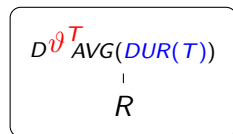
$$D^{\vartheta^T}_{AVG}(DUR(T))(\epsilon_U(R))$$

2. Timestamp substitution:

$$D^{\vartheta^T}_{AVG}(DUR(U))(\epsilon_U(R))$$

3. Temporal adjustment:

$$\mathcal{N}_D(\epsilon_U(R), \epsilon_U(R))$$



Answering Sequenced Queries/2

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

1. Timestamp propagation:

$$D^{\vartheta^T}_{AVG}(DUR(T))(\epsilon_U(R))$$

2. Timestamp substitution:

$$D^{\vartheta^T}_{AVG}(DUR(U))(\epsilon_U(R))$$

3. Temporal adjustment:

$$\mathcal{N}_D(\epsilon_U(R), \epsilon_U(R))$$

4. Nontemporal aggregation:

$$D, T^{\vartheta}_{AVG}(DUR(U))(\mathcal{N}_D(\epsilon_U(R), \epsilon_U(R)))$$

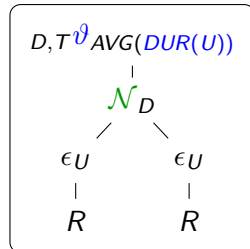
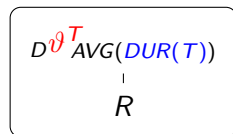
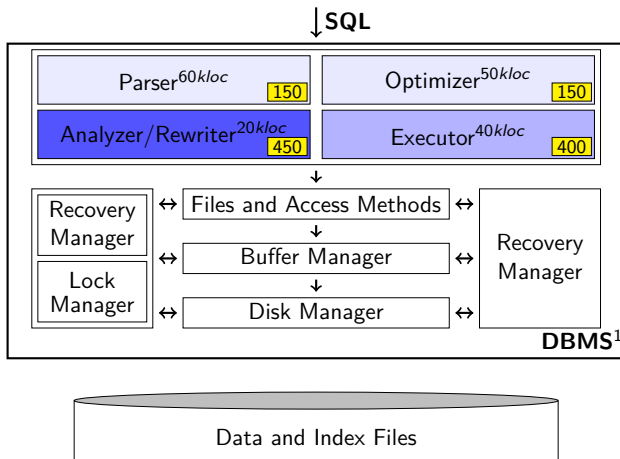


Table of Contents

- ▶ Elements of the sequenced semantics
 - ▶ Snapshot reducibility
 - ▶ Change preservation
 - ▶ Extended snapshot reducibility
- ▶ Integration of temporal support in DBMS kernel
 - ▶ Algebraic adjustment operators
 - ▶ Reduction rules
- ▶ **SQL implementation**

PostgreSQL Implementation/1

- DBMS kernel integration of temporal adjustment.



- SQL extension that provides direct access to the adjustment operators:

$\epsilon_U(R)$: **SELECT** Ts **AS** Us, Te **AS** Ue, ***** **FROM** R

Handwritten notes:
- "gold copies of - TS and TE" with a red circle around "Ts AS Us, Te AS Ue"
- "take all attributes" with a red arrow pointing to the asterisk (*)

$\mathcal{N}_B(R, S)$: **FROM** (R **NORMALIZE** S **USING** (B)
 WITH (Rs, Re, Ss, Se)) **AS** R

$\phi_\theta(R, S)$: **FROM** (R **ALIGN** S **ON** θ
 WITH (Rs, Re, Ss, Se)) **AS** R

- Additional (technical) details will be added on the following slides
- Source Code: <http://tpg.inf.unibz.it>

Timestamp Propagation in SQL

- ▶ The **extend** operator adds to the schema of a relation R an attribute U that is a duplicate of the period of R .
- ▶ Algebra: $\epsilon_U(R)$
- ▶ SQL:

WITH

R AS (SELECT *, Ts AS Us, Te AS Ue FROM R)

...

Temporal Normalization in SQL

- ▶ The **normalize** operator splits each tuple in R with respect to the tuples in S that match on the grouping attributes B .
- ▶ Algebra: $\mathcal{N}_B(R, S)$ or $\mathcal{N}_\theta(R, S)$
- ▶ SQL:

```
... FROM (R NORMALIZE S
          USING (B)
          WITH (Rs, Re, Ss, Se)) AS Rnorm ...
```

```
... FROM (R NORMALIZE S
          ON  $\theta$ 
          WITH (Rs, Re, Ss, Se)) AS Rnorm ...
```

- ▶ In SQL the timestamp attributes must be specified explicitly

Temporal Alignment in SQL

- ▶ The **align** operator splits each tuple in R with respect to each tuple in S that satisfies θ .
- ▶ Algebra: $\phi_{\theta}(R, S)$
- ▶ SQL:

```
FROM ( R ALIGN S
      ON  $\theta$ 
      WITH (Rs, Re, Ss, Se) ) AS Radj
```

- ▶ In SQL the timestamp attributes must be specified explicitly

Example/1

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

R

$r_1 = (\text{Ann})$	
$r_2 = (\text{Joe})$	$r_3 = (\text{Ann})$

$\epsilon_U(R)$

$r_1 = (\text{Ann}, [1, 8])$	
$r_2 = (\text{Joe}, [2, 6])$	$r_3 = (\text{Ann}, [8, 12])$

$\mathcal{N}(\epsilon_U(R), \epsilon_U(R))$

$(\text{Ann}, [1, 8])$	
$(\text{Ann}, [1, 8])$	$(\text{Ann}, [1, 8])$
$(\text{Joe}, [2, 6])$	$(\text{Ann}, [8, 12])$

$T^{\vartheta}_{AVG}(DUR(U))(\mathcal{N}(\epsilon_U(R), \epsilon_U(R)))$

(7)	(5.5)	(7)	(4)
-----	-------	-----	-----

Translate resulting algebra expression to SQL:

► $\tau_{\text{AVG}(\text{DUR}(U))}(\mathcal{N}(\epsilon_U(R), \epsilon_U(R)))$

► Schema: $R(N, Ts, Te)$

```
► WITH x AS ( SELECT Ts Us, Te Ue, * FROM R )
  SELECT AVG(Ue - Us), Ts, Te
  FROM (x AS x1 NORMALIZE x AS x2
        USING()
        WITH (Ts,Te,Ts,Te)) AS Rnorm
  GROUP BY Ts, Te;
```

Summary

- ▶ Sequenced semantics
 - ▶ Snapshot reducibility
 - ▶ Change preservation
 - ▶ Extended snapshot reducibility
- ▶ New algebraic adjustment operators in the DBMS kernel that provide support for the sequenced semantics.
 - ▶ normalize
 - ▶ align
 - ▶ extend
- ▶ Timestamp propagation
- ▶ Direct mapping of the adjustment operators to SQL for illustration purposes. Other SQL extensions are possible.