**Goal and Requirements**

**Goal**: Scaling of attribute values when temporal operators modify attached timestamps.

For instance, attribute values that record total quantities over time often must be scaled if the associated timestamp changes.

**Example**: A budget of 181K during [Feb, Aug] corresponds to a budget of 89K during [Feb, May].

**Application Requirements**: Applications have the choice to specify whether and how to scale attribute values at query time.

---

**Solution**

**Solution**: 
- An algebra that performs adjustment of timestamps automatically and allows to scale attribute values.
- User-defined functions that scale attribute values based on:  
  - the value $x$ to be scaled;  
  - the new and old timestamp

**Challenge**: 
- Old values are required for scaling  
- Old timestamps are required for scaling  
- New timestamps are required for scaling

---

**Example**

**Input**: Project $N$ with budget $B$ in department $D$ during time $T$.  
**Query**: How does the available budget per department change?

<table>
<thead>
<tr>
<th>Project</th>
<th>Budget</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>181K</td>
<td>Feb-Aug</td>
</tr>
<tr>
<td>$N$</td>
<td>89K</td>
<td>Feb-May</td>
</tr>
</tbody>
</table>

**Example SQL mapping**:

```sql
SELECT ABSORB * FROM r
FROM (r ALIGN s ON
  $p_x = \sigma_T(p)$
  $p_y = \sigma_T(p)$
) p

\[ D^T \sum(B) \]  

**Example RA mapping**:

\[ D^T \sum(B) \equiv D^T \sum(B) (N_{p_1 \to p_2} (\epsilon_U \eta_U (p_1, p_2))) \]

**Example SQL mapping**:

```sql
WITH  
px AS (SELECT Ts Us, Te Ue, * FROM p)  
SELECT D, SUM(B), Ts, Te  
FROM (px p1 NORMALIZE px p2 ON p1.D=p2.D) p
GROUP BY D, Ts, Te
```

---

**Background**

### Temporal Algebra:

- **Operator**: Reduction  
- **Function**:  
  - $\sigma_T(p)$  
  - $\pi_T(p)$  
  - $\rho_T(p)$  
  - $\rho_{B}(p)$  
  - $\rho_{N}(p)$  

### SQL mapping:

- **Operator**:  
- **SQL**:  
  - $\sigma_T(n)$: SELECT Ts Us, Te Ue, * FROM p  
  - $\rho_{N}(p)$: FROM (p NORMALIZE n ON (r)) WHERE  
  - $\rho_{B}(p)$: FROM (p ALIGN s ON (r)) WHERE  

---

**Scaling Functions**

**Definition**: Scales an attribute value $x$ from an old timestamp $T_{old}$ to a new timestamp $T_{new}$:

\[ scale(x, T_{new}, T_{old}) \rightarrow x' \]

**Example**: UDF-function scaling uniformly in PostgreSQL’s procedural language PL/pgSQL:

```sql
CREATE FUNCTION scale(X float, ts_new DATE, ts_old DATE, T_new DATE, T_old DATE)  
RETURNS float AS $$
BEGIN  
RETURN X * (ts_new - ts_old) / (ts_old - ts_old);  
END; $$ LANGUAGE PL/pgSQL;
```

**Other scaling functions**:  
- Considering working days only  
- Atomic attribute values  
- Trends 

---

**Published in**: *Proceedings of the 29th IEEE International Conference on Data Engineering (Demo)*, Brisbane, Australia, April 8-11, 2013.