Querying Temporal Databases and Data Streams

Miguel Ceriani\textsuperscript{1} \hspace{1cm} Szymon Klarman\textsuperscript{2} \hspace{1cm} Evgeny Sherkhonov\textsuperscript{3}

(1) University of Rome, Italy  
(2) Centre for Artificial Intelligence Research, South Africa  
(3) University of Amsterdam, The Netherlands

January 30  
FCCOD 2014, Bolzano
We want to extend the basic relational data model and develop methods and tools to be able:

- to represent when data are true (validity & transaction time),
- to query data taking into account this temporal information.

References:

A relational database is a first-order structure over a (finite) data domain $D$ and a schema $\rho = (r_1, \ldots, r_k)$, consisting of a set of relations $(r_1, \ldots, r_k)$. A tuple $r_i(\vec{a})$ is true in an instance of $(D, \rho)$ iff $\vec{a} \in r_i^D$.

<table>
<thead>
<tr>
<th>name</th>
<th>department</th>
</tr>
</thead>
<tbody>
<tr>
<td>john</td>
<td>d1</td>
</tr>
<tr>
<td>mark</td>
<td>d2</td>
</tr>
</tbody>
</table>

Definition (Temporal domain)

A temporal domain $T_P$ is a tuple $(T, <)$, where $T$ is a nonempty set of elements called time instants and $<$ is an irreflexive, linear ordering on $T$. 

M. Ceriani, S. Klarman, E. Sherkhonov
Temporal data model

Definition (Timestamp model)

A timestamp TDB is a first-order structure

\[ D \cup T_P \cup (R_1, \ldots, R_k), \]

where \( R_i \) are temporal relations-instances of the temporal extensions \( R_i \) of \( r_i \), where:

\[ R_i(t, \vec{a}), \text{ for some } t \in T_P, \text{ iff } r_i(\vec{a}). \]

<table>
<thead>
<tr>
<th>time</th>
<th>name</th>
<th>department</th>
</tr>
</thead>
<tbody>
<tr>
<td>1999</td>
<td>john</td>
<td>d1</td>
</tr>
<tr>
<td>2000</td>
<td>john</td>
<td>d1</td>
</tr>
<tr>
<td>2001</td>
<td>john</td>
<td>d3</td>
</tr>
<tr>
<td>2002</td>
<td>mark</td>
<td>d2</td>
</tr>
<tr>
<td>2001</td>
<td>mark</td>
<td>d2</td>
</tr>
</tbody>
</table>
Temporal data model

Definition (Snapshot model)

A snapshot TDB over $D$, $T_P$, and $\rho$, is a map $DB : T_P \mapsto DB(D, \rho)$, where $DB(D, \rho)$ is the class of (finite) relational databases over $D$ and $\rho$.

<table>
<thead>
<tr>
<th>Year</th>
<th>Emp</th>
<th>name</th>
<th>department</th>
</tr>
</thead>
<tbody>
<tr>
<td>1999</td>
<td>Emp</td>
<td>john</td>
<td>d1</td>
</tr>
<tr>
<td>2001</td>
<td>Emp</td>
<td>john</td>
<td>d3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>mark</td>
<td>d2</td>
</tr>
</tbody>
</table>
There exists a direct correspondence between the timestamp and the snapshot models:

\[ \forall t \in T. \forall a_1, \ldots, a_k \in D : (r^{D(t)}_i(a_1, \ldots, a_k) \leftrightarrow R^D_i(t, a_1, \ldots, a_k)) \]

TDBs expressed in the timestamp and snapshot models are called *abstract*. 
Abstract query languages

The most natural languages for querying abstract TDBs are variants of FOL over the vocabulary $(=, r_1, \ldots, r_k)$ of the extended structure:

- **two-sorted FOL** (the timestamp model)
- **FO temporal logic** (the snapshot model)
The syntax of the two-sorted FO language $L^P$:

$$M ::= R_i(t_i, \vec{x}) \mid t_i < t_j \mid x_i = x_j \mid \neg M \mid M \land M \mid \exists t_i. M \mid \exists x_i.M$$

where $R_i$ is the temporal extension of $r_i$, for $r_i \in \rho$. Variables $t_i$ range over $T$ and $x_i$ over $D$.

Example:

$$\exists x_2.(Emp(t_0, x_1, x_2) \land \exists t_1.(t_0 < t_1 \land \exists x_3.(Emp(t_1, x_1, x_3) \land \neg (x_2 = x_3))))$$

Answers: $t_0 \mapsto 1999$, $x_1 \mapsto john$ and $t_0 \mapsto 2000$, $x_1 \mapsto john$.  

M. Ceriani, S. Klarman, E. Sherkhonov
FO temporal logic

**Temporal operators syntax:**

\[ O ::= t_i < t_j \mid \neg O \mid O \land O \mid \exists t_i. O \mid X_i \]

where \( X_i \) are propositional variables. An \( n \)-ary temporal operator is an \( O \)-formula with exactly one free variable \( t_0 \) and \( n \) free propositional variables \( X_1, \ldots X_n \). A set of temporal connectives is denoted by \( \Omega \).

**Examples:**

\[
\begin{align*}
\text{always-in-future}(X) & \triangleq \forall t_1.(t_0 < t_1 \rightarrow X(t_1)) \\
\text{sometime-in-future}(X) & \triangleq \exists t_1.(t_0 < t_1 \land X(t_1)) \\
\vdots & \triangleq \vdots
\end{align*}
\]
The syntax of the FO temporal language $L^\Omega$:

$$F ::= r_i(\vec{x}) \mid x_i = x_j \mid \neg F \mid F \land F \mid \omega(F_1, \ldots, F_n) \mid \exists x_i . F$$

where $r \in \rho$ and $\omega$ is an $n$-ary temporal operator.

Example:

$$\exists x_2.(Emp(x_1, x_2) \land \text{ sometime-in-future}(\exists x_3.(Emp(x_1, x_3) \land \neg(x_2 = x_3))))$$

Answer: $x_1 \mapsto john$ in 1999, 2000.
Expressive power

There exists a translation from $L^\Omega$ to $L^P$, hence $L^\Omega \subseteq L^P$.

**Theorem (Abiteboul et al., 1996)**

$L_{\text{since, until}} \sqsubseteq L^P$ over the class of finite timestamp TDBs.

**Theorem (Toman, Niwinski, Bidoit et al.)**

$L^\Omega \sqsubset L^P$ over the class of timestamp TDBs for an arbitrary finite set of first-order temporal connectives $\Omega$.

Observation: $L^\Omega$ cannot express query “are there two distinct time instants at which a unary relation $R$ contains exactly the same values?”. In $L^P$:

$$\exists t_1, t_2. (t_1 < t_2 \land \forall x. (R(t_1, x) \iff R(t_2, x)))$$
Concrete databases

Abstract TDBs can be in principle infinite but should be representable in a finite form. *Concrete* TDBs are these finite representations.

**Definition (Interval-based temporal domain)**

Let $T_P = (T, \lt)$ be a discrete linearly ordered point-based temporal domain. We define the set:

$$I(T) = \{ (a, b) : a \leq b, a \in T \cup \{-\infty\}, b \in T \cup \{\infty\} \}.$$

*Interval-based temporal domain* is the structure $T_I = (I(T), <_{--}, <_{+-}, <_{-+}, <_{++})$, where $<_{--}, <_{+-}, <_{-+}, <_{++}$ express ordering relationships over $I(T)$. 

M. Ceriani, S. Klarman, E. Sherkhonov

Querying Temporal Databases and Data Streams
Concrete databases

**Definition (Concrete TDB)**

A concrete TDB is a finite first-order structure $D \cup T_I \cup \{R_1, \ldots, R_k\}$, where $R_i$ are the concrete temporal relations which are finite instances of $R_i$ over $D$ and $T_I$.

<table>
<thead>
<tr>
<th>Emp</th>
<th>time</th>
<th>name</th>
<th>department</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[1999, 2000]</td>
<td>john</td>
<td>d1</td>
</tr>
<tr>
<td></td>
<td>[2000, 2001]</td>
<td>mark</td>
<td>d2</td>
</tr>
</tbody>
</table>

**Definition (Semantic Mapping $\parallel \cdot \parallel$)**

Let $D_1$ be an abstract TDB and $D_2$ a concrete TDB over the same schema $\rho$. $D_2$ encodes $D_1$ (written $\parallel D_2 \parallel = D_1$) if:

$$R^D_1(t, x) \iff \exists I \in T_I. R^D_2(I, x) \land t \in I$$
Abstract vs. concrete TDBs

Abstract Temporal Databases

\[ D: \{(a, 1), (a, 2), (a, 3), \ldots \} \]

\[ \varphi = R(x, y) \land y < 10 \]

eval(\(\varphi\))(\(E_1\))

\[ \{(a, [1, 5]), (a, [3, \infty])\} \]

Concrete Temporal Databases

eval(\(\varphi\))(\(E_2\))

\[ \{(a, [1, \infty])\} \]

\[ \varphi(D): \{(a, 1), (a, 2), \ldots, (a, 9)\} \]

\[ \{(a, [1, 9])\} \]
Concrete temporal query language

The syntax of the interval-based language $L^I$:

$$M ::= R_i(I, \vec{x}) \mid I_i^* < I_j^* \mid x_i = x_j \mid \neg M \mid M \land M \mid \exists I_i . M \mid \exists x_i . M$$

where $R_i$ is the temporal extension of $r_i$, for $r_i \in \rho$, and $I_i^* \in \{I^+, I^−\}$

Example:
For databases $D_1, D_2$ and the relation $r \in \rho$, such that $R_{D_1} = \{([1, 2], a), ([1, 3], a)\}$ and $R_{D_2} = \{([1, 3], a)\}$:

$$\exists I, J. \exists x . (R(I, x) \land R(J, x) \land I \neq J)$$

Answer:
$x \mapsto a$ in $D_1$, and $\emptyset$ in $D_2$. 

M. Ceriani, S. Klarman, E. Sherkhonov

Querying Temporal Databases and Data Streams
Definition ($\| \cdot \|$-generic queries)

Let $\| \cdot \|$ be the semantic mapping and $\varphi \in L^I$. We say that $\varphi$ is $\| \cdot \|$-generic if $\|D_1\| = \|D_2\|$ implies $\|\varphi(D_1)\| = \|\varphi(D_2)\|$ for all concrete TDBs $D_1, D_2$. 

The challenge is then to devise methods ensuring that querying remains $\| \cdot \|$-generic.
One solution is to use *compilation* techniques, i.e., transform $T^P$ queries into $L^I$ while preserving meaning under $\|\cdot\|$. 

**Theorem (Toman 1996)**

There is a (recursive) mapping $F : L^P \leftrightarrow L^I$ such that $\varphi(\|D\|) = \|F(\varphi)(D)\|$. 

**Theorem (Toman 1996)**

For every $\|\cdot\|$-generic $\varphi \in L^I$ there is $\psi \in L^P$ such that $\|\varphi(D)\| = \psi(\|D\|)$ for all concrete temporal databases $D$. 
Temporal extensions of SQL

Challenges for practical temporal extensions

- Multi-set (bag) semantics of SQL,
- Extensions must support the chosen model of time,
- Efficient query evaluation over concrete databases.

The majority of extensions assume point based semantics but use syntax based on intervals (Allen’s interval algebra),

Extensions based on: abstract and concrete query languages.
Based on abstract language $L^p$: SQL/TP

- Simple extension of SQL with data type based on the point-based temporal domain,
- Bag semantics,
- Can be efficiently evaluated via translation of $L^p$ to $L^I$,

```sql
select r1.name
from Emp r1, Emp r2
where r1.name = r2.name
  and r1.time < r2.time
  and not exists ( select *
      from Emp r3
      where r3.name = r1.name
      and r1.time < r3.time
      and r3.time < r2.time )
```

Based on abstract language $L^\Omega$

- Added similarly to set operators.
  
  \[ Q_1 \text{ until } Q_2 \quad Q_1 \text{ since } Q_2 \]

- A natural extension of ATSQL’s sequenced semantics [Snodgrass et al., 1995],

Two ways of evaluating:

- Over coalesced concrete databases using the translation from $L^\Omega$ to $L^I$;
- By composing the translation of $L^\Omega$ to $L^p$ with translation to $L^p$ to $L^I$. 
A lot of proposals

- SQL/Temporal,
- AT-SQL,
- Temporal extension of Informix.

Syntax is extended with Allen’s interval algebra expressions,
Multi-set semantics.
Example for SQL/Temporal

```
select r1.name
from Emp r1, Emp r2
where r1.name = r2.name
    and r1.time before r2.time
```

Incorrect! Reason: non-generic.

Two approaches to overcome this:

- Coalescing. → incompatible with bag semantics
- Folding and Unfolding. → space blow-up
Further extensions

- Beyond first order logic.
  - Extended with monadic quantifiers over temporal domain,
  - Fixpoints.
- Beyond Closed World Assumption.
  - Quickly leads to undecidability even in append-only databases.
  - Decidable fragments: monadic temporal extensions, temporal logic programs.
Temporal databases

Data streams

Foundations

Further challenges

Updating temporal databases

- Insertion is easy for both abstract and concrete databases.
- Deletion and update is not straightforward for concrete databases.

Example

Assume DB contains a tuple \([1999, 2005], \text{john}, d1\). We want to specify that john was sacked in 2001 but was hired back to the same department in 2003.

- Delete \([1999, 2005], \text{john}, d1\),
- Add tuples \([1999, 2001], \text{john}, d1\) and \([2003, 2005], \text{john}, d1\).
Updates in append-only database, expiration and stream

- An update adds a new state to the existing finite history,
- Expiration techniques are needed for forgetting old data: administrative and query-driven approaches
- This is very similar to the problem of efficient data storing in data streams
- Continuous queries in data streams are similar to queries over database histories.
Data Streams

Different Applications
- Sensor Networks (smart homes, smart cities)
- Social Data
- Network Traffic Analysis
- Financial Tickers
- ...

Common Requirements
- Input stream(s) unbounded in space and time (only a small portion of data available at a time)
- Timely reaction is needed, i.e. continuous queries
- Order and rate of data arrival is not under control of the system
“Kind of” the Temporal DBs *timestamp model*, but...

- Time has a linear discrete model
- Temporal Relations have a finite encoding, while Data Streams may be *infinite*
- Temporal Queries are usually one-time, while Data Stream Queries are typically *continuous*
- Temporal Queries may be unbounded in time, while Data Stream Queries are typically on *windows*
Sliding Windows

Window

Past Data

Recent Data

Future Data

M. Ceriani, S. Klarman, E. Sherkhonov

Querying Temporal Databases and Data Streams
Tumbling Windows

- Window w1: Events E at time 2 and 4
- Window w2: Event E at time 7
- Window w3: Event E at time 10

Hopping Windows

- Window w1: Events E at time 2 and 4
- Window w2: Event E at time 6
- Window w3: Event E at time 10
- Window w4: Event E at time 13
- Window w5: Event E at time 15
Continuous Query Processing Model

- Input/Outputs are Streams or (Temporal) Relations
- Network (DAG) of operators:
  - **Stream-to-relation** operators:
    - *Now* operator
    - *Time-based Sliding/Tumbling Window* operator
  - **Relation-to-relation** operators:
    - *Relational Algebra* operators
  - **Relational-to-stream** operators:
    - *Insert Stream*
    - *Delete Stream*
    - *Relation Stream*
An Example: Linear Road

Input: stream of positions and speeds of vehicles.
Output: the tolls for vehicles.
Linear Road: Network of Queries Used

- TollStr
- SegVolRel
- CongestedSegRel
- VehicleSegEntryStr
- ActiveVehicleSegRel
- SegSpeedStr
- PosSpeedStr

M. Ceriani, S. Klarman, E. Sherkhonov
CREATE VIEW CongestedSegRel(segNo) AS

SELECT segNo
FROM SegSpeedStr [RANGE 5 MINUTES]
GROUP BY segNo
HAVING AVG(speed) < 40
CREATE VIEW TollStr (vehicleId, toll) AS

SELECT RSTREAM(E.vehicleId, 
    2 * (V.numVehicles - 50) * (V.numVehicles - 50) 
AS toll) 
FROM VehicleSegEntryStr [NOW] AS E, 
    CongestedSegRel AS C, 
    SegVolRel AS V 
WHERE E.segNo = C.segNo AND C.segNo = V.segNo
Asynchronous processing: \texttt{update/computeAnswer}

Bounded Space (dealing with)
- Avoid Unneeded Materialization
- Synopsis Data Structures
- Sketches (approximation of synopsis)

Bounded Time (dealing with)
- Incremental Evaluation
- Batch Processing (slow \texttt{computeAnswer})
- Sampling (slow \texttt{update})
Challenging Domains

- Streaming of Social Data
- Streaming and the Semantic Web
- Stream Monitoring
- XML Streams
- Uncertain Streams
- Streaming Frameworks and Systems
- Distributed Streams
Additional References

- Michael Benedikt, Dan Olteanu: **Report on the first Workshop on Innovative Querying of Streams**. SIGMOD Record, June 2013 (Vol. 42, No. 2)