Tagging Stream Data for Rich Real-Time Services

Rimma V. Nehme
Elke A. Rundensteiner
Elisa Bertino

Presented by: Shujie Li
13. 01. 2010
Contents

- **Introduction**
  - Approach
  - Fundamentals
- **Tag Model**
- **Tag Query Language (TAG-QL)**
  - Key Statements
  - Attach/Generate a tag to objects
- **Tag-Base Query Processing**
  - Tag-Oriented Query Processing
  - Tag-Aware Query Processing
- **Experimental Analysis**
- **Conclusion**
Contents

- Introduction
  - Approach
  - Fundamentals
    - Tag Model
    - Tag Query Language (TAG-QL)
      - Key Statements
        - Attach/Generate a tag to objects
    - Tag-Base Query Processing
      - Tag-Oriented Query Processing
      - Tag-Aware Query Processing
    - Experimental Analysis
  - Conclusion
Introduction: Approach

Tag: Accident; 2 Cars; Near exit 12.

Query: Retrieve all Tags within a certain radius

Accident or road construction??

Why stopped??

Accident!!!!
● Premise of tagging: Users can label data in order to get more informative query results.

● The additional label with type of metadata

  Tick-tags

● Continuous query processing with tags, address the Tick-tag issues and efficiency concerns.
Proposed Solution: Stream Tag Framework (STF)

Data Model:

tuple = [ stream_id, tuple_id, A, timestamp ]
Introduction: Fundamental

Data Stream Management System (DSMS):
The database for managing continuous data streams which are sequences of data tuples.

Tagging:
The process of adding comments or labels to something.

Tagging in Data Stream Environments:
Additional information to streaming objects (tuples, tuples attribute, etc.).

Streaming objects  Tagged stream

\[ o_3 \rightarrow o_2 \rightarrow o_1 \rightarrow o_3 \rightarrow t_3 \rightarrow o_2 \rightarrow t_2 \rightarrow o_1 \rightarrow t_1 \]
Contents

- Introduction
  Approach
  Fundamentals
- **Tag Model**
  - Tag Query Language (TAG-QL)
    Key Statements
    Attach/Generate a tag to objects
  - Tag-Base Query Processing
    Tag-Oriented Query Processing
    Tag-Aware Query Processing
  - Experimental Analysis
  - Conclusion
Tag Model: Fundamental

- **Definition:**
  Meta-Data tuples that attach additional information to stream objects.

- **Characteristics:**
  transient, sequential access, high input rate, potentially infinite size, continuous tag processing.
Tag Model: Design

**TID:** Unique id of the tagger / user, determined by the system.
Applicability:

Describes the stream object, regular expression.
Content:

A string datatype, stores the actual tag value. E.g. “Accident”.
Type:

To classify streaming tags: Objective type (i.e. “2 Car Accident”), Subjective type (i.e. “Nice”, “Interesting”), etc…
Sign:

To serve as a qualitative description of a tick-tag based on the content to generate an overall opinion for the tagged information.

“+”: Positive content; “-”: Negative content.
Lifespan:

A time interval in which the tick-tag is active.

Exception “I” (Instant): if a single applicability is wanted.
Mode:

Indicates the user’s preference regarding the combination of the actual tag with earlier ones.

“O”: Overwrite; “C”: Combine.

Point: Tagger specifies only self tags.
**Timestamp:**

The time when the tick-tag was generated.
**Example:**

**Auction Stream contains items to sell**

**Schema:**

<table>
<thead>
<tr>
<th>Seller_id</th>
<th>Product</th>
<th>Product feature</th>
<th>St_price</th>
<th>time</th>
</tr>
</thead>
</table>

**Example:**

| 123       | Dell Laptop | pink, 1420     | 600 Euro | 2:00 pm |

**TAG1:** with respect to **VALUE** of start price (**St_price**).

```
<table>
<thead>
<tr>
<th>TID</th>
<th>Stream(s), Tuple(s), Attribute(s)</th>
<th>Blah ...</th>
<th>1 day</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>,</strong>, {St_price. value}</td>
<td>Fair</td>
<td>+</td>
<td></td>
</tr>
</tbody>
</table>

Value is given by system
```
Contents

- Introduction
  Approach
  Fundamentals
- Tag Model
- Tag Query Language (TAG-QL)
  Key Statements
  Attach/Generate a tag to objects
- Tag-Base Query Processing
  Tag-Oriented Query Processing
  Tag-Aware Query Processing
- Experimental Analysis
- Conclusion
## Tag Query Language: Key Statements

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATTACH TAG...</td>
<td>Attaches a tag to a streaming object</td>
</tr>
<tr>
<td>SELECT TAGS...</td>
<td>Selects tags that satisfy a certain search predicate</td>
</tr>
<tr>
<td>SELECT TAGGED OBJECTS...</td>
<td>Selects tagged objects</td>
</tr>
<tr>
<td>SELECT... WITH TAGS</td>
<td>Returns tag-enriched query results</td>
</tr>
</tbody>
</table>
Tag Query Language: Attach a tag to objects

How to attach a tick-tag to a streaming object?

Method 1: manually attaching

Syntax:

```
ATTACH TAG <tag_content>
TO <object_description>
(WHERE <condition_description>)
(WITH
  TAG_SIGH = <+ | - >
  TAG_LIFESPAN = <lifespan_value>
  TAG_MODE = <mode_value>)
```

Indicates the object to which the tag should be attached

Decides the location of the tag
Example 1:
Tag with respect to VALUE of St_price

ATTACH TAG 'Fair'
TO Auction . St_price . Value
WITH

TAG_SIGH = '+' AND
TAG_LIFESPAN = 1 day AND
TAG_MODE = OVERWRITE

---

| TID | Stream(s), Tuple(s), Attribute(s) | Blah ... | + | 1 day | O |
How to attach a tick-tag to a streaming object?

Method 2: continuous attaching

Example 2:

Tag with respect to VALUE of Seller_id

```
ATTACH TAG 'Expensive' CONTINUOUSLY TO Auction.Seller_id.value WHERE (SELECT Seller_id FROM Auction WHERE St_price > 600) WITH TAG_SIGH = '－'
```

Keyword: Tagging is continuous

Continuous adding tick-tag to the seller id’s value of auction with St_price > 600.
Contents

- Introduction
  Approach
  Fundamentals
- Tag Model
- Tag Query Language (TAG-QL)
  Key Statements
  Attach/Generate a tag to objects
- Tag-Base Query Processing
  Tag-Oriented Query Processing
  Tag-Aware Query Processing
- Experimental Analysis
- Conclusion
Tag-Based Query Processing

Category:

- Tag-Oriented Query Processing (TOQ Processing):
  - Users query tick-tags *explicitly*

- Tag-Aware Query Processing (TAQ Processing):
  - Users query tick-tags *implicitly*
Tag-Oriented Query Processing

Expressing in TAG-QL:

Q1: Tags where the tags values are of interest (‘select tags’)

Q2: Tags where the corresponding base data values are of interest (‘select tagged object’)

<table>
<thead>
<tr>
<th>pid</th>
<th>measure</th>
<th>location</th>
<th>time</th>
</tr>
</thead>
</table>

Patient Stream:

Q1: SELECT TAGS

FROM Patient

WHERE OBJECT = Patient.measure AND

TAG_SIGN = ‘-’

Q2: SELECT TAGGED OBJECT

FROM Patient

WHERE TAG = `Emergency`
Tag-Oriented Query Algebra:

Tagger Operator:

Input: a stream of objects & Output: a stream of objects with an inserted tag $t$

$$[TO(O, p_o, t) \rightarrow O'] \quad \text{with} \quad \forall t_i \in T', t_i = t$$

$O_2$ satisfies $p_o$
Tag-Oriented Query Algebra:

Tag Selection:

Input: a stream objects with tags    Output: a stream of tags

\[
\left[ TS \left( \overrightarrow{O}, p_t \right) \rightarrow T' \right] \quad \text{with} \quad T' \subseteq T
\]

Search predicate on tag

t_1 \text{ and } t_2 \text{ satisfy } p_t
Tag-Oriented Query Algebra:

Tagged Object Selection:

Input: a stream of objects with tags  
Output: stream of objects

\[ [TOS (\mathcal{O}, p_t) \rightarrow O'] \text{ with } O' \subseteq O \]

\[ t_1 \text{ and } t_2 \text{ satisfy } p_t \]
Tag-Oriented Query Algebra:

Tag Join:

\[ TJ \left( O_1, O_2, E \right) \rightarrow O' \]  with  \( T' = E(T_1, T_2) \neq \emptyset \)

E: Some tag Join condition, i.e., if the both tags are equivalence, or if the both have the same meaning.

\( T_1 \) and \( T_2 \) join based on \( E \)

Streaming Tag Index (stix)
Tag-Oriented Query Algebra:

Tag-Based Aggregation:

\[ TG \left( \overrightarrow{O}, E, G_{T}^{agg} \right) \rightarrow \overrightarrow{O}'_{G_{T}^{agg}} \]

Some tag join condition

A certain aggregate function

Streaming Tag Index (stix)

State buffer

Aggregated value

\( t_1 \) and \( t_2 \) “are the same” based on E
Tag-Aware Query Processing

Goals Tag-Aware Query:
Returns continuous query results that are “enriched” with the tags attached to the original base data.
i.e. enriched tuples / tagged data tuples

Idea:
with statement “WITH TAGS”.
Expressing in TAG-QL:

Patient Stream:

```
| pid | measure | location | time |
```

Q3:  
```
SELECT  pid, location, time  
FROM    Patient  
WHERE   measure > 80  
WITH    TAGS
```
Tag-Aware Query Algebra:

Projection operator:

Process tuples by extracting wanted attributes.

Propagates tick-tags and thereafter the projected tuples.

Discard the tick-tag which is attached to the projected attributes.
Example:

Stream $R_1$

Data schema

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
</tr>
</tbody>
</table>

Relation Algebra Projection:

$\Pi_{B,C}(R_1) = \begin{array}{cc}
B & C \\
\hline
b & c \\
\end{array}$

Tag-Aware Query Algebra:

SELECT $B, C$
FROM $R_1$
Tag-Aware Query Algebra:

Selection operator:

Drops tuples that do not satisfy the selection condition.

Propagation of tags delayed until min. one tagged tuple which fulfills the selection condition found.

If all tagged tuples are filtered then their corresponding tag is discarded.
Tag-Aware Query Algebra:

Aggregation operator:

Each attribute domain is split into attribute sub-groups which contain the tuples with the same attribute value.
Tag-Aware Query Algebra:

Aggregation operator:

Calculate a result for each sub-group.

Sends the result to the output stream preceded by the collection of tags which are applicable to any object in that sub-group.
Tag-Aware Query Algebra:

Join operator:

Tuples x and y join

Produces a new joined tuple from x and y

Stream Tag Index

(1:1, 4-5)_2 (1:1-3)_1
Contents

- Introduction
  - Approach
  - Fundamentals
- Tag Model
- Tag Query Language (TAG-QL)
  - Key Statements
  - Attach/Generate a tag to objects
- Tag-Base Query Processing
  - Tag-Oriented Query Processing
  - Tag-Aware Query Processing
- Experimental Analysis
- Conclusion
Experimental Analysis

Setup:

- **Stream Tag Framework** is implemented in a DSMS prototype **CAPE**.
- Data generated by the **Network-based Moving Objects Generator**.
- **100K** of moving objects, which present **cars, cyclists, pedestrians**.
- The moving objects stream are broken up into **several streams** based on the **id of objects**.
Four Types of queries are used:
Comparison:

Tagger Operator VS. regular Selection Operator

Tagger Operator is used!!

Tagger Operator is larger than regular Selection Operator !!!
Comparison:

**Tick-Tag VS Alternative Tagging Approach**

**Alternative Tagging Approach:**

- **Table Approach**
  - Produce a separate global table which maintains all tags.

- **Extended Data Tuples**
  - Extend the data tuple by adding an attribute for tag information.

- **Streaming XML**
  - Dynamic data which is in an XML format
Comparison:

**Tick-Tag** VS. **Alternative Tagging Approach**

Tick-Tag approach produces higher output rates!!!
Comparison:

Tick-Tag VS. Alternative Tagging Approach

Tick-Tag approach produces smaller memory usage !!!
Cost of Tag Join Operator:

Tag Join is used!!

More tags, more overhead !!!!
Comparison:

Tag-Aware Join VS. regular Join Operator

Tag-Aware Join is used!!
Contents

- Introduction
  - Approach
  - Fundamentals
- Tag Model
- Tag Query Language (TAG-QL)
  - Attach a tag to an object
  - Tag-Based Query Processing
  - Tag-Oriented Query Processing (TOQP)
  - Tag-Aware Query Processing (TAQP)
- Experimental Analysis
- Conclusion
Conclusion

- Propose the flexible STF to support for tagging data stream, and where the Tick-tags are attached to the objects.

- Tag Query Language enable attachment and query of streaming tags.

- Tag-Based Query Processing contains two aspects.

- Experiment shows the scalability and benefits of Tick-tags in contrast to the traditional theory.
Thank You