Workshop on Big (and Small) Data in Science and Humanities @ BTW 2019

Temporal Graph Analysis using Gradoop

5th March 2018

Christopher Rost
Leipzig University

Prof. Dr. Andreas Thor
University of Applied Sciences for Telecommunications Leipzig

Prof. Dr. Erhard Rahm
Leipzig University
MOTIVATION

- Call center network of 25 banks of The Banks Association of Turkey
- ~ 7,500 agents
- ~ 46 million incoming calls answered by agents per month
- ~ 24 million total outbound calls to customers per month
- ~ 24 million active customers per month
- 16 service types (card, stock, ATM, online banking, …)

Source: The Banks Association of Turkey Call Center Statistics December 2017
PROPERTY GRAPH

Nodes represent entities

Edges connect nodes and represent relationships

Nodes can have an id, type label and properties as K/V pairs

Edges are directed and can have an id, type label and properties as K/V pairs

[1] Agent
- Agent_id: 4242
- Service: stock
- Location: Istanbul
- Sex: female
- Age: 32

[2] Call
- At: 2017-02-05 14:35:24
- Duration: 240s

[1] Call
- At: 2017-02-05 14:35:24
- Duration: 240s

[2] Call
- At: 2017-02-06 12:15:00
- Duration: 125s

[2] Customer
- Customer_id: 1234
- Name: Bob
- Country: GER
- City: Berlin
- CreatedAt: 2016-12-01
SOME ANALYTICAL QUESTIONS

- How is the average talk time of incoming calls of the investment line service per month in 2017?

- How the average speed of answers changed over the year 2018?

- Which customers call the same service multiple times a day?

- Which customers did agent Alice call on March, 2018? What was the maximum, minimum and average call time?
MOTIVATION

- Most real-world networks evolve over time
- Graph elements are continuously added, removed or updated
- Analytical questions are often time related
- Most graph processing systems focus on static graphs

⇒ Scalable graph processing system to analyze temporal dimensions
REQUIREMENTS

WHAT DO WE NEED?

- Scalable temporal graph processing system
- Flexible bitemporal graph model
- Support timestamps, time-intervals and non-temporal graph elements
- Graph operators, e.g., snapshot retrieval, graph evolution, temporal grouping, subgraph extraction, pattern matching
- Chain operators to build temporal analysis workflows
THE GRADOOP SYSTEM

- Open Source framework for distributed, declarative graph analytics
- Support of heterogeneous graphs and collections of those
- Composable graph operators and algorithms via GrALa

⇒ www.gradoop.com

High-level architecture of Gradoop [Ju18]
TEMPORAL PROPERTY GRAPH MODEL (TPGM)

// extends EPGM

- Added four obligatoric time attributes \((val\text{-}from, val\text{-}to), (tx\text{-}from, tx\text{-}to)\)
- Times can be (1) empty, (2) a timestamp or (3) a time-interval
  - Flexible representation, also edge-centric scenarios can be modeled
- Valid times are the responsibility of the user
- Transaction times can be maintained by the system
- Whole graph with rollback and historical information
- Chaining of operators \(\rightarrow\) analytical workflow
TPGM EXAMPLE (1)

**Agent**
- **Agent_id**: 4242
- **Service**: stock
- **Location**: Istanbul
- **Sex**: female
- **Age**: 32

**Call**
- **val-from**: 2017-02-05 14:35:24
- **val-to**: 2017-02-05 14:39:24
- **tx-from**: 2017-04-20 13:34:00
- **tx-to**: 9999-12-31 23:59:59

**Customer**
- **Customer_id**: 1234
- **Name**: Bob
- **Country**: GER
- **City**: Berlin

**Call**
- **val-from**: 2017-02-06 12:15:00
- **val-to**: 2017-02-06 12:17:05
- **tx-from**: 2017-04-20 13:34:01
- **tx-to**: 9999-12-31 23:59:59

**Call**
- **val-from**: 2016-12-01 00:00:00
- **val-to**: -
- **tx-from**: 2017-02-20 12:30:00
- **tx-to**: 9999-12-31 23:59:59
TPGM EXAMPLE (2)

Agent

Name: Alice
Service: Stock
Location: Istanbul

[1] Call
val-from: 2017-02-05 14:35:24
val-to: 2017-02-05 14:39:24
tx-from: 2017-04-20 13:34:00
tx-to: 9999-12-31 23:59:59

[2] Customer
val-from: 2016-12-01 00:00:00
val-to: -
tx-from: 2017-02-20 12:30:00
tx-to: 9999-12-31 23:59:59
Customer_id: 1234
Name: Bob
Country: GER
City: Berlin

[2] Call
val-from: 2017-02-06 12:15:00
val-to: 2017-02-06 12:17:05
tx-from: 2017-04-20 13:34:01
tx-to: 9999-12-31 23:59:59
TPGM EXAMPLE (3)

[1] Call
val-from: 2017-02-05 14:35:24
val-to: 2017-02-05 14:39:24
tx-from: 2017-04-20 13:34:00
tx-to: 9999-12-31 23:59:59

[2] Call
val-from: 2017-02-06 12:15:00
val-to: 2017-02-06 12:17:05
tx-from: 2017-04-20 13:34:01
tx-to: 9999-12-31 23:59:59

Agent
Name: Alice
Service: Stock
Location: Istanbul

Customer
[1, -]
Name: Bob
Location: Leipzig
TPGM EXAMPLE (4)

Agent

Name: Alice
Service: Stock
Location: Istanbul

Call

[5, 6]

Call

[7, 10]

Customer

[1, -]
Name: Bob
Location: Leipzig
TPGM EXAMPLE (5)

Agent
Name: Alice
Service: Stock
Location: Istanbul

Call [2, 4]

Customer
Name: Andy
Location: Berlin

Call [5, 6]

Agent
Name: Brat
Service: Stock
Location: Istanbul

Call [1, 5]

Customer
Name: Bob
Location: Leipzig

Call [8, 10]

Agent
Name: Chris
Service: ATM
Location: Istanbul

Call [5, 6]

Customer
Name: Carol
Location: Berlin
Mail: carol@examp.le

Call [3, 5]

Customer
Name: Dave
Location: Munich
Gender: male

Call [7, 10]
OPERATORS

Transformation

Grouping

Snapshot

Graph Evolution
OPERATORS

Transformation

Grouping

Snapshot

Graph Evolution
TRANSFORMATION

Graph = Graph.transform(graphFunction, vertexFunction, edgeFunction)

- Structure preserving modification of graph elements
- Pre-defined and user-defined transformation functions
  - Modification of temporal attributes
  - Fill temporal attributes from property data
  - Create properties from temporal information

```java
graph.transform(
    g -> g,
    v -> v,
    e -> {e['Duration'] = e.to - e.from})
```
OPERATORS

Transformation

Grouping

Snapshot

Graph Evolution
SNAPSHOT

Graph = Graph.snapshot(temporalPredicateFunction)

- Temporal analysis might focus on the state of a graph
  - At a specific point in time
  - For a given time range
- Implies the extraction of a subgraph
- Vertex- and Edge-induced snapshots are supported
- Predefined predicate functions available
  - Adopted from SQL:2011 standard (temporal databases)
  - AS OF, FROM … TO … , BETWEEN … AND …
GraphAsOf2 = Graph.snapshot(AsOf('2'))
GraphAsOf2 = Graph.snapshot(AsOf('2'))
OPERATORS

Transformation   Grouping

Snapshot   Graph Evolution
GRAPH EVOLUTION

Graph = Graph.diff(firstTempPredicate, secondTempPredicate)

- Evolution of a graph can be represented as difference between snapshots

- Results in a graph with annotated elements ...
  - Added elements
  - Deleted elements
  - Persistent elements
GraphDiff = Graph.diff(AsOf('2'), AsOf('6'))
GraphDiff = Graph.diff(AsOf('2'), AsOf('6'))

Agent
Name: Alice
Service: Stock
Location: Istanbul

Agent
Name: Brat
Service: Stock
Location: Istanbul

Agent
Name: Chris
Service: ATM
Location: Istanbul

Customer
Name: Andy
Location: Berlin

Customer
Name: Bob
Location: Leipzig

Customer
Name: Carol
Location: Berlin
Mail: carol@examp.le
GraphDiff = Graph.diff(AsOf('2'), AsOf('6'))
TEMPORAL GRAPH ANALYSIS USING GRADOOP | Workshop BigDS @ BTW 2019

GraphDiff = Graph.diff(AsOf('2'), AsOf('6'))

---

Agent
[ ]
Name: Alice
Service: Stock
Location: Istanbul

Call
[2, 4]

Call
[5, 6]

Call
[6, 8]

Agent
[ ]
Name: Brat
Service: Stock
Location: Istanbul

Customer
[1, -]
Name: Andy
Location: Berlin

Mail: carol@examp.le

Customer
[1, -]
Name: Bob
Location: Leipzig

Customer
[3, -]
Name: Dave
Location: Munich
Gender: male

Customer
[2, -]
Name: Carol
Location: Berlin

Agent
[ ]
Name: Chris
Service: ATM
Location: Istanbul

Call
[5, 6]
OPERATORS

Transformation

Grouping

Snapshot

Graph Evolution
GROUPING

Graph = Graph.groupBy(verGrpKeys, verAggF, edGrKeys, edAggF)

- Structural grouping based on labels and attributes
- Three additional features to EPGM-Grouping
  - Time-specific value transformation functions, e.g., Year(), Day(), …
  - GROUP BY CUBE, GROUP BY ROLL UP
  - Pre-defined time-specific aggregation functions, e.g., MinFrom(), AvgDuration(), …

How long are customers talking with agents on average by location and service?
GROUPING

Graph = Graph.groupBy(verGrpKeys, verAggF, edGrKeys, edAggF)

How long are customers talking with agents on average by location and service?

GraphCollection = Graph.groupBy(
    [':label', 'Location', 'Service'] BY ROLLUP,
    [superVertex['count'] = Count()],
    [':label'],
    [superEdge['avg'] = AvgDuration()]
)
TEMPORAL GRAPH ANALYSIS USING GRADOOP | Workshop BigDS @ BTW 2019

[‘:label’, ‘Location’, ‘Service’] BY ROLLUP

Agent

Name: Alice
Service: Stock
Location: Istanbul

Call
[2, 4]

Agent

Name: Brat
Service: Stock
Location: Istanbul

Call
[1, 5]

Agent

Name: Chris
Service: ATM
Location: Istanbul

Call
[5, 6]

Customer

[1, -]
Name: Andy
Location: Berlin

Call
[5, 6]

Customer

[1, -]
Name: Bob
Location: Leipzig

Call
[6, 8]

Customer

[2, -]
Name: Carol
Location: Berlin
Mail: carol@examp.le

Call
[3, 5]

Customer

[3, -]
Name: Dave
Location: Munich
Gender: male

Call
[7, 10]
TEMPORAL GRAPH ANALYSIS USING GRADOOP | Workshop BigDS @ BTW 2019

\[
\begin{array}{l}
\text{:label, 'Location', 'Service'} \quad \text{BY ROLLUP}
\end{array}
\]

\[
\begin{array}{l}
\text{Agent} \\
\quad \text{Service: Stock} \\
\quad \text{Location: Istanbul} \\
\quad \text{count: 2}
\end{array}
\]

\[
\begin{array}{l}
\text{Call} \\
\quad [1, 10] \\
\quad \text{avg: 2.66}
\end{array}
\]

\[
\begin{array}{l}
\text{Agent} \\
\quad \text{Service: ATM} \\
\quad \text{Location: Istanbul} \\
\quad \text{count: 1}
\end{array}
\]

\[
\begin{array}{l}
\text{Call} \\
\quad [2, 6] \\
\quad \text{avg: 1.5}
\end{array}
\]

\[
\begin{array}{l}
\text{Call} \\
\quad [3, 6] \\
\quad \text{avg: 1.5}
\end{array}
\]

\[
\begin{array}{l}
\text{Call} \\
\quad [7, 10] \\
\quad \text{avg: 3}
\end{array}
\]

\[
\begin{array}{l}
\text{Customer} \\
\quad [1, -] \\
\quad \text{Location: Leipzig} \\
\quad \text{count: 1}
\end{array}
\]

\[
\begin{array}{l}
\text{Customer} \\
\quad [3, -] \\
\quad \text{Location: Munich} \\
\quad \text{count: 1}
\end{array}
\]

\[
\begin{array}{l}
\text{Customer} \\
\quad [1, -] \\
\quad \text{Location: Berlin} \\
\quad \text{count: 2}
\end{array}
\]

\[
\begin{array}{l}
\text{Customer} \\
\quad [1, -] \\
\quad \text{Location: Munich} \\
\quad \text{count: 1}
\end{array}
\]

\[
\begin{array}{l}
\text{Call} \\
\quad [2, 6] \\
\quad \text{avg: 1.5}
\end{array}
\]

\[
\begin{array}{l}
\text{Call} \\
\quad [7, 10] \\
\quad \text{avg: 3}
\end{array}
\]

\[
\begin{array}{l}
\text{Call} \\
\quad [1, 10] \\
\quad \text{avg: 2.75}
\end{array}
\]

\[
\begin{array}{l}
\text{Agent} \\
\quad \text{count: 3}
\end{array}
\]

\[
\begin{array}{l}
\text{Agent} \\
\quad \text{count: 3}
\end{array}
\]

\[
\begin{array}{l}
\text{Agent} \\
\quad \text{count: 3}
\end{array}
\]

\[
\begin{array}{l}
\text{Agent} \\
\quad \text{count: 3}
\end{array}
\]

\[
\begin{array}{l}
\text{Agent} \\
\quad \text{count: 3}
\end{array}
\]

\[
\begin{array}{l}
\text{Agent} \\
\quad \text{count: 3}
\end{array}
\]

\[
\begin{array}{l}
\text{Agent} \\
\quad \text{count: 3}
\end{array}
\]

\[
\begin{array}{l}
\text{Agent} \\
\quad \text{count: 3}
\end{array}
\]

\[
\begin{array}{l}
\text{Agent} \\
\quad \text{count: 3}
\end{array}
\]

\[
\begin{array}{l}
\text{Agent} \\
\quad \text{count: 3}
\end{array}
\]

\[
\begin{array}{l}
\text{Agent} \\
\quad \text{count: 3}
\end{array}
\]

\[
\begin{array}{l}
\text{Agent} \\
\quad \text{count: 3}
\end{array}
\]

\[
\begin{array}{l}
\text{Agent} \\
\quad \text{count: 3}
\end{array}
\]

\[
\begin{array}{l}
\text{Agent} \\
\quad \text{count: 3}
\end{array}
\]

\[
\begin{array}{l}
\text{Agent} \\
\quad \text{count: 3}
\end{array}
\]

\[
\begin{array}{l}
\text{Agent} \\
\quad \text{count: 3}
\end{array}
\]

\[
\begin{array}{l}
\text{Agent} \\
\quad \text{count: 3}
\end{array}
\]

\[
\begin{array}{l}
\text{Agent} \\
\quad \text{count: 3}
\end{array}
\]
CONCLUSION AND FUTURE DIRECTIONS

- TPGM with bitemporal time support
- Operators for temporal analysis workflows
- Integration into distributed graph analysis system Gradoop
- Complete implementation
- Operator optimization
- Graph stream support

Visit gradoop @ www.gradoop.com