UNIVERSITÄT LEIPZIG

Europäische Union
Europäischer Fonds für regionale Entwicklung
Europäischer Sozialfonds

Europa fördert Sachsen.
EFRE
ESF

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

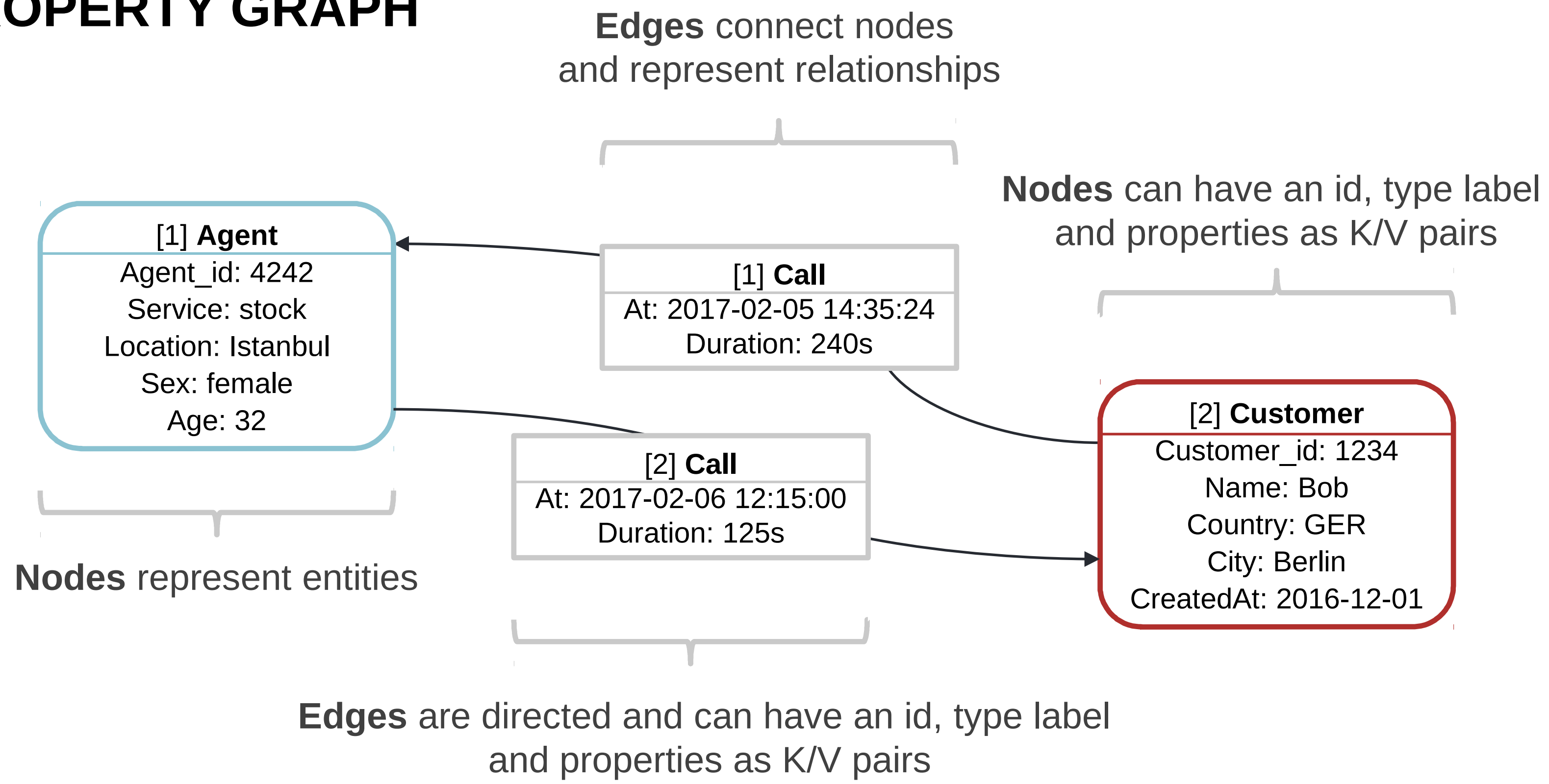
Prof. Dr. Erhard Rahm
Leipzig University

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



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?

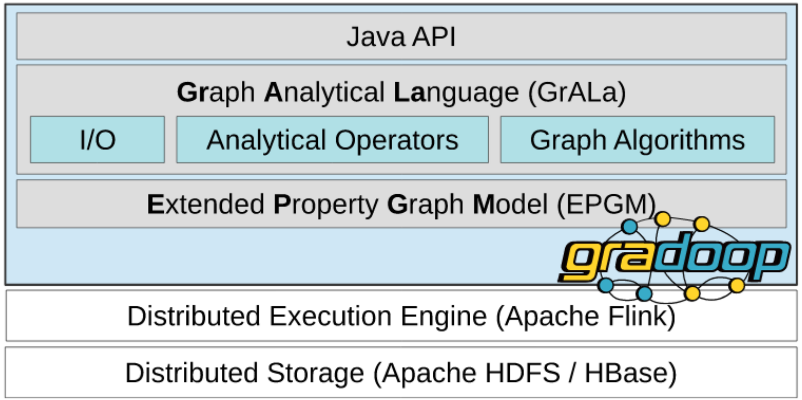
- Scalable temporal graph processing system
- Flexible bitemporal graph model
- Support timestamps, time-intervals and non-temporal graph elements
- Graph operators, e. g., snapshot retrieval, graph evolution, temporal grouping, subgraph extraction, pattern matching
- Chain operators to build temporal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

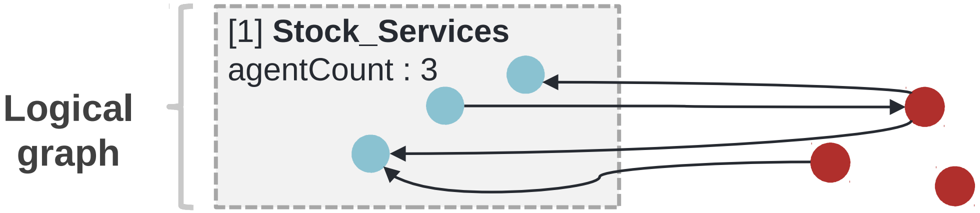
THE GRADOOP SYSTEM

- 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]



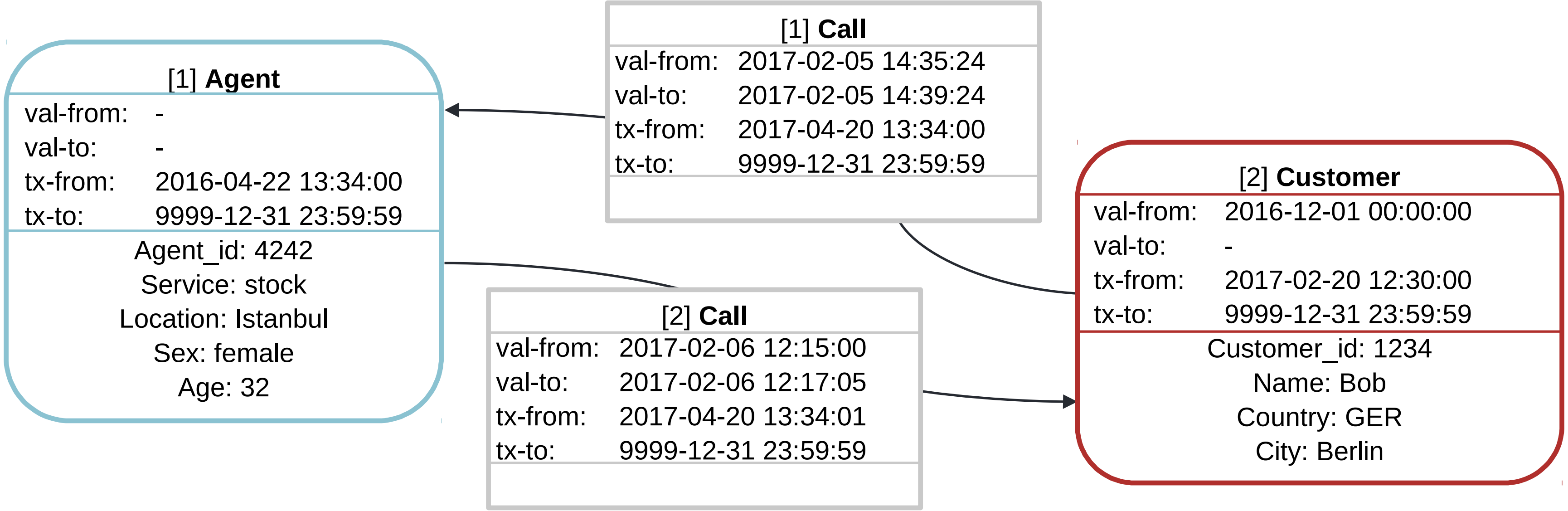


TEMPORAL PROPERTY GRAPH MODEL (TPGM)

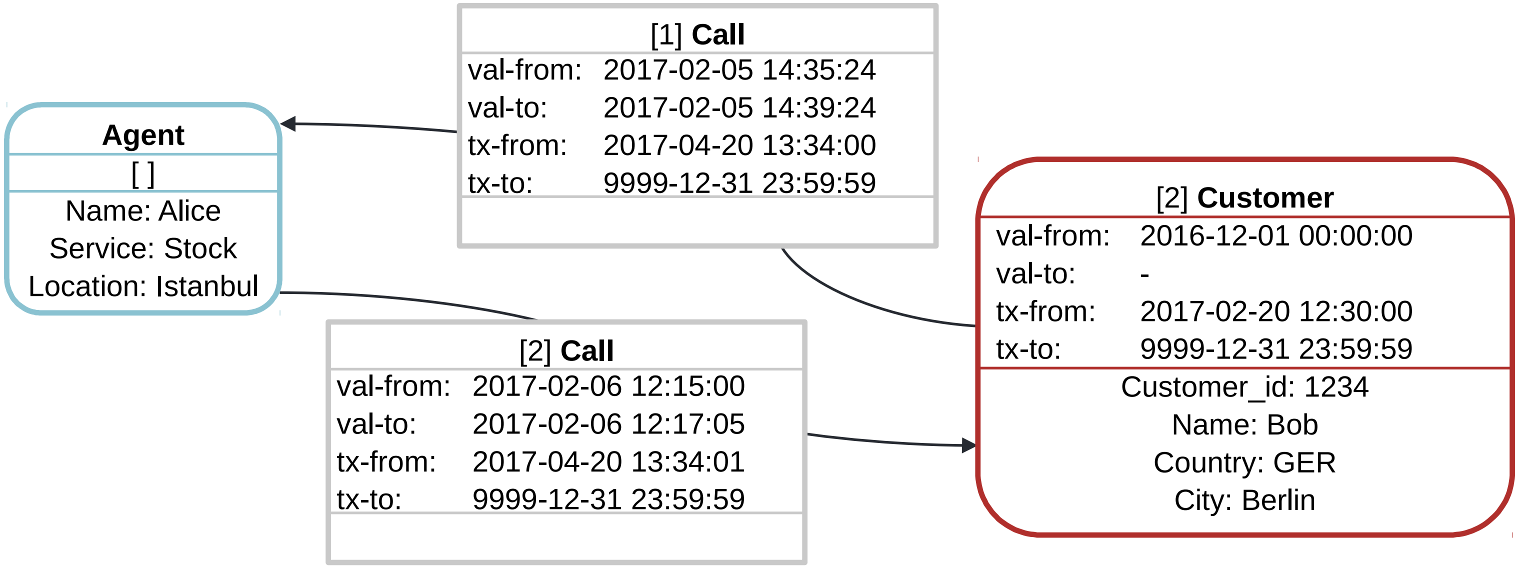
```
// extends EPGM
```

- Added four obligatoric time attributes *(val-from, val-to), (tx-from, tx-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 → analytical wokflow

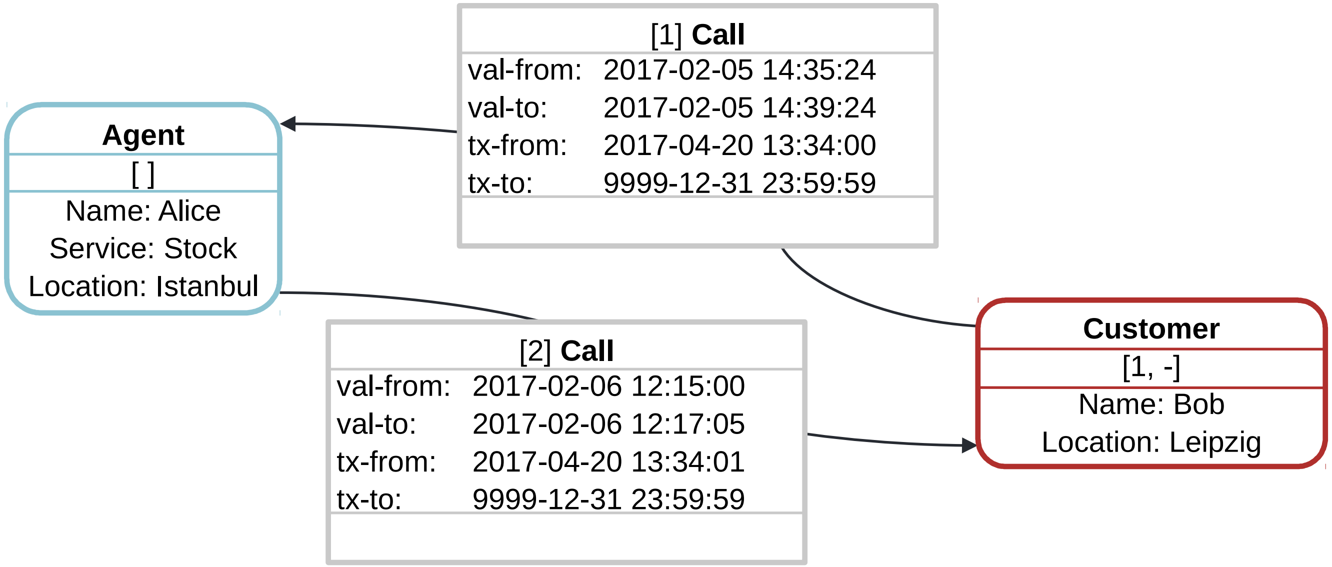
TPGM EXAMPLE (1)



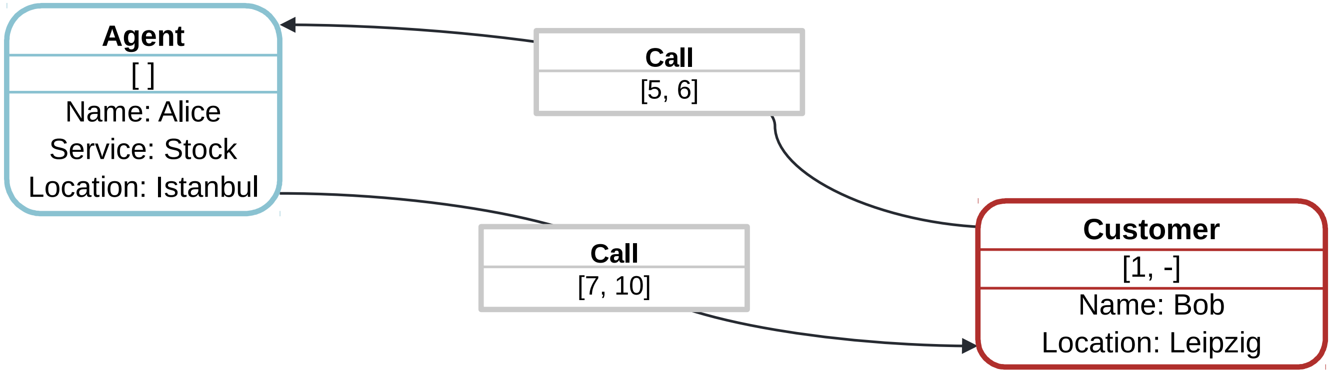
TPGM EXAMPLE (2)



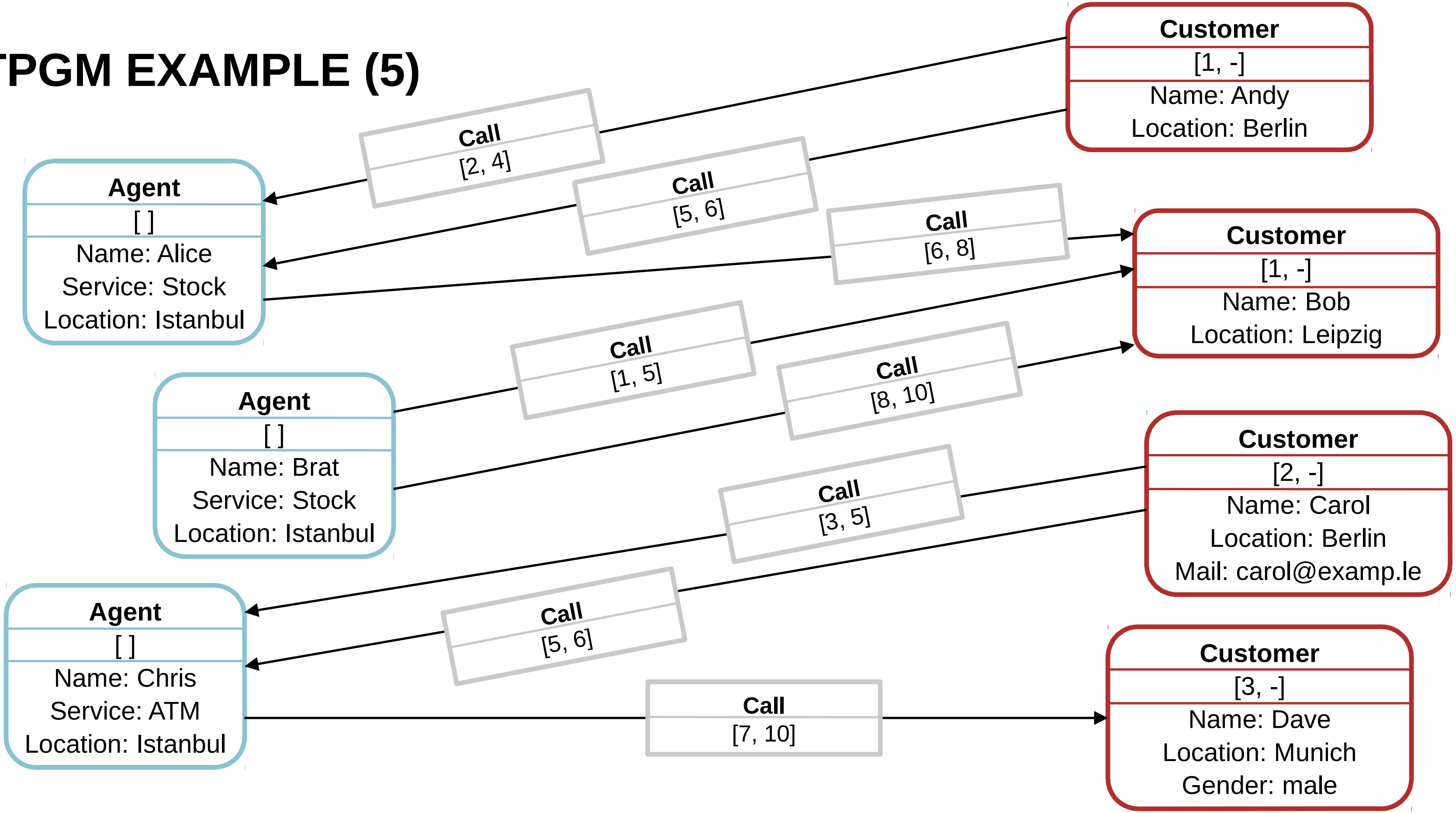
TPGM EXAMPLE (3)



TPGM EXAMPLE (4)



TPGM EXAMPLE (5)



OPERATORS

Transformation	Grouping
Snapshot	Graph Evolution

EPGM

TPGM

OPERATORS

Transformation	Grouping
Snapshot	Graph Evolution

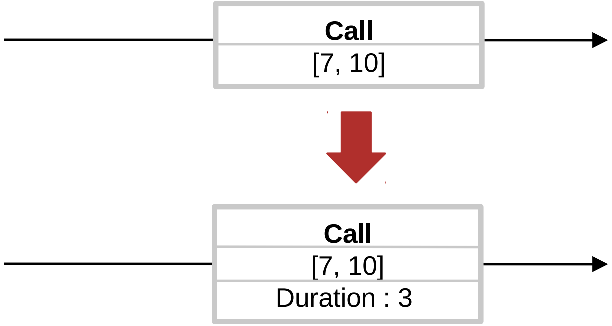
EPGM
TPGM

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

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



OPERATORS

Transformation	Grouping
Snapshot	Graph Evolution

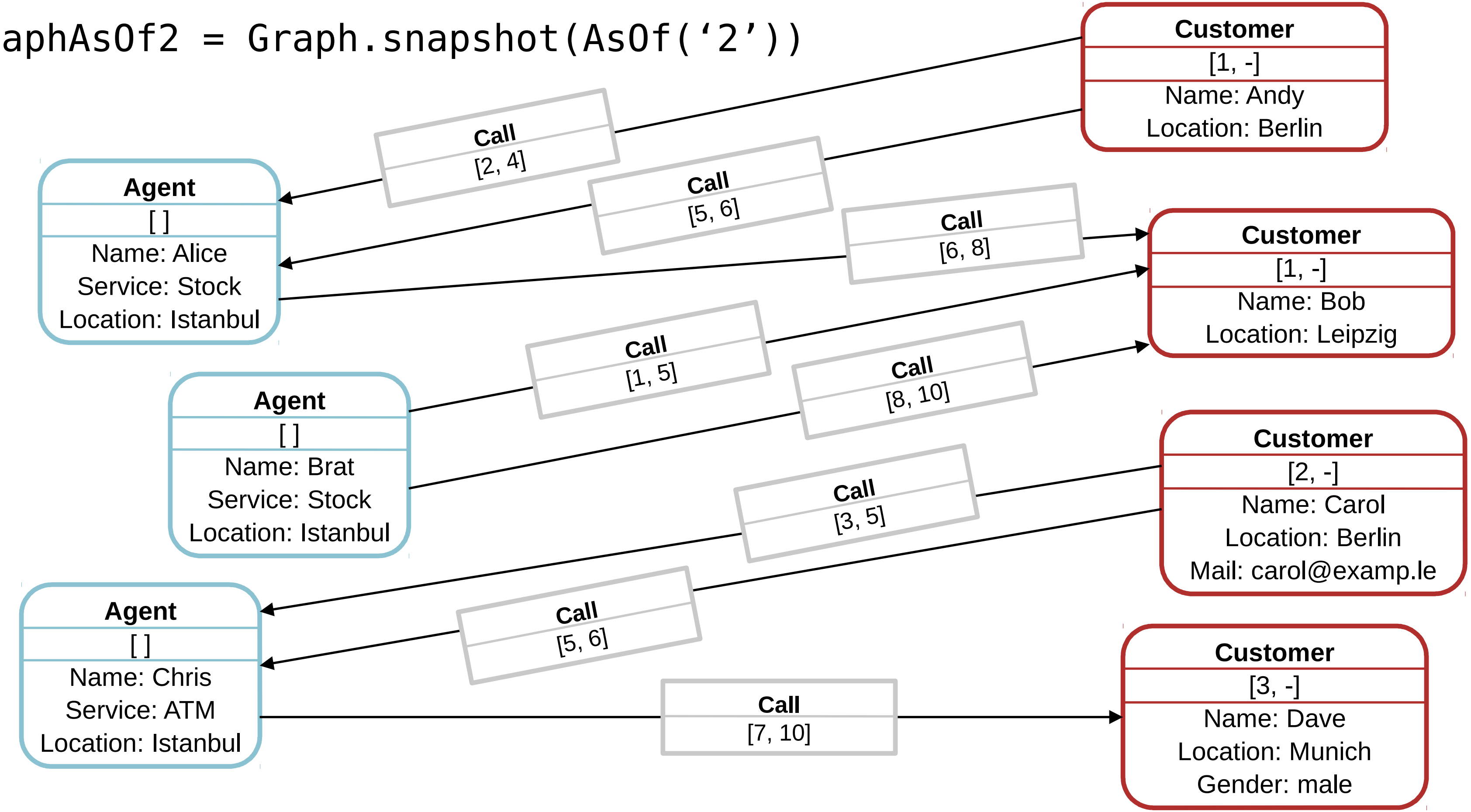
EPGM
TPGM

SNAPSHOT

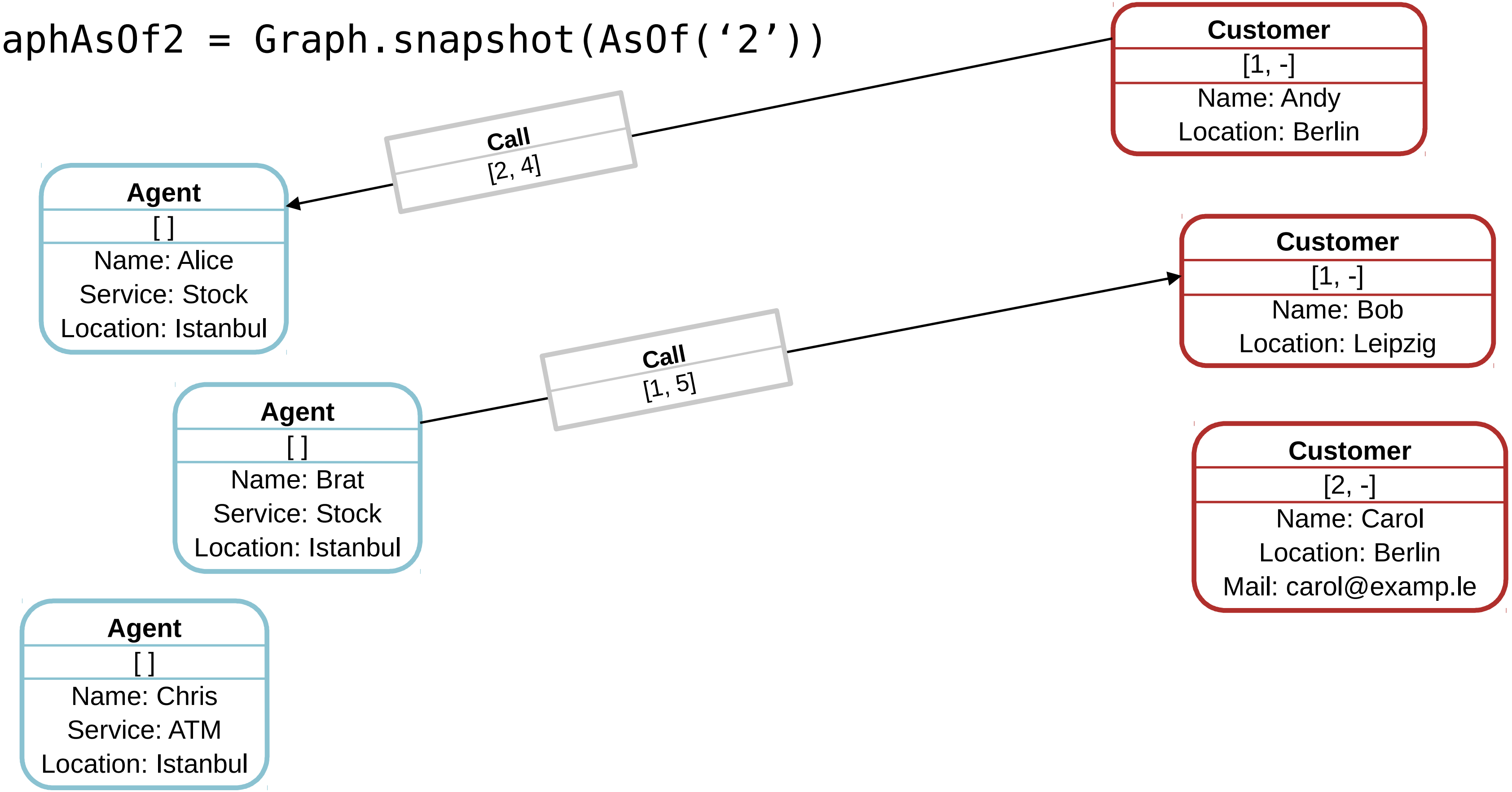
```
Graph = Graph.snapshot(temporalPredicateFunction)
```

- 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

EPGM

TPGM

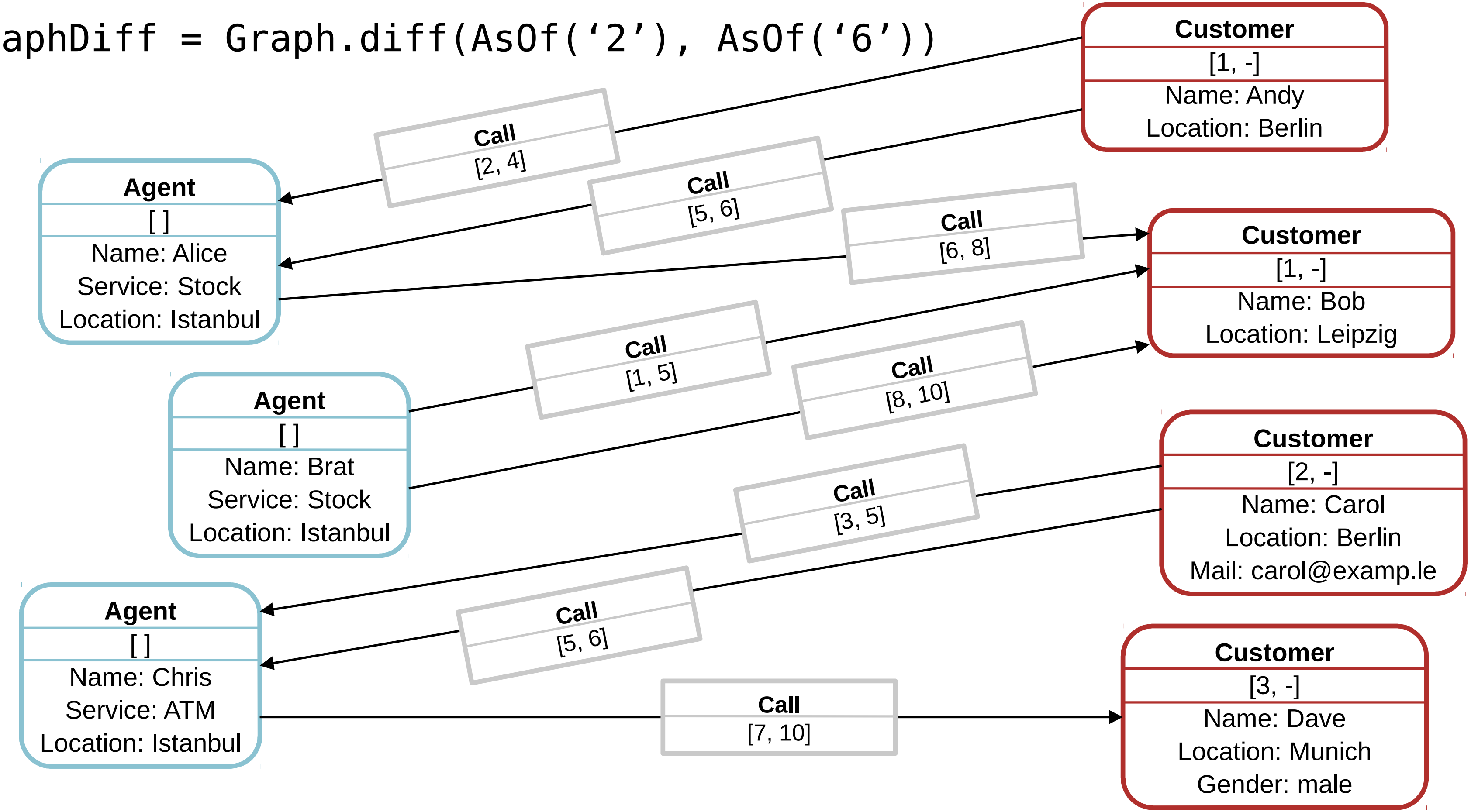
GRAPH EVOLUTION

```
Graph = Graph.diff(firstTempPredicate, secondTempPredicate)
```

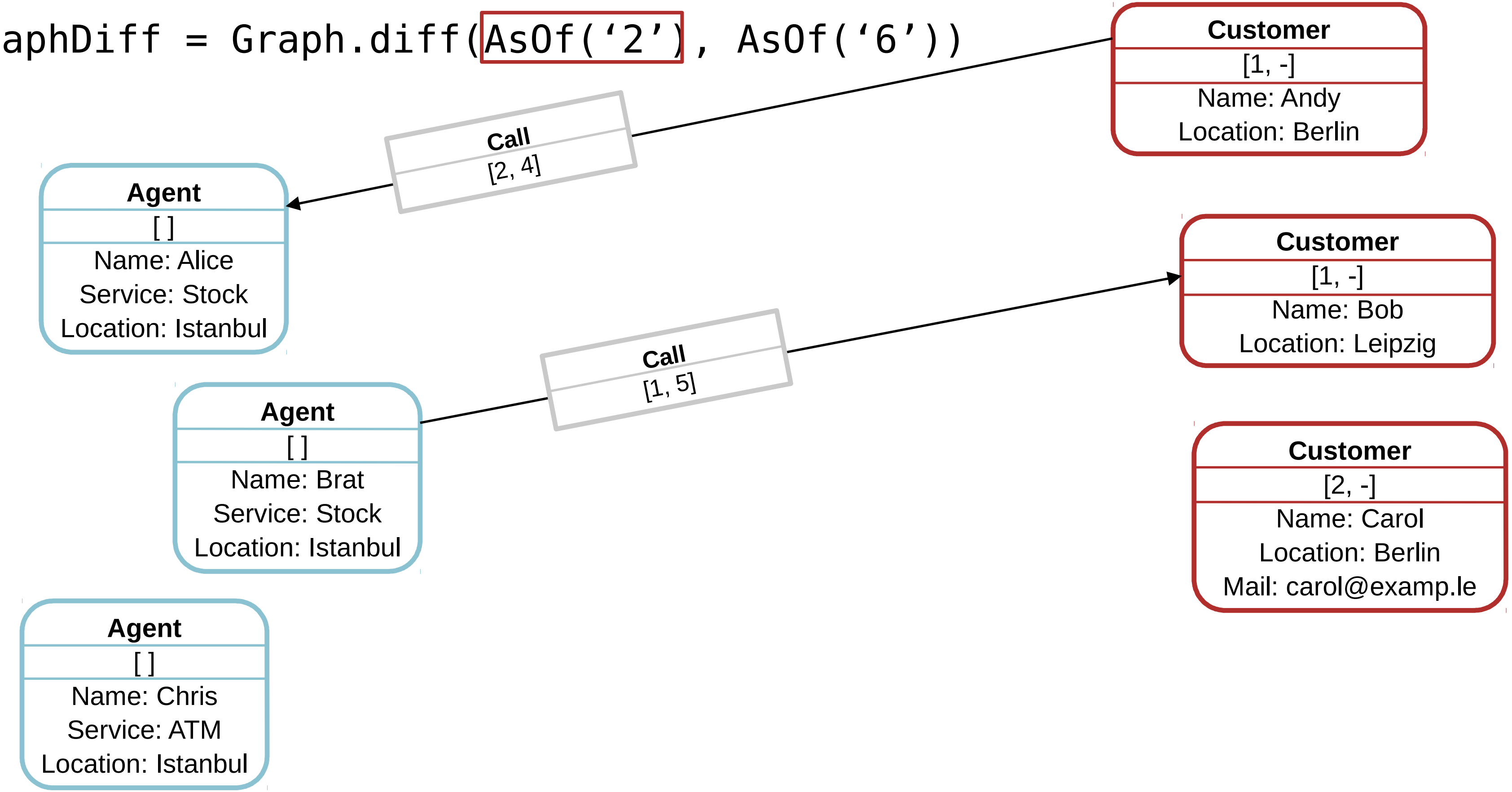
- Evolution of a graph can be represented as difference between snapshots

- Results in a graph with annotated elements …
  - Added elements +
  - Deleted elements −
  - Persistent elements =

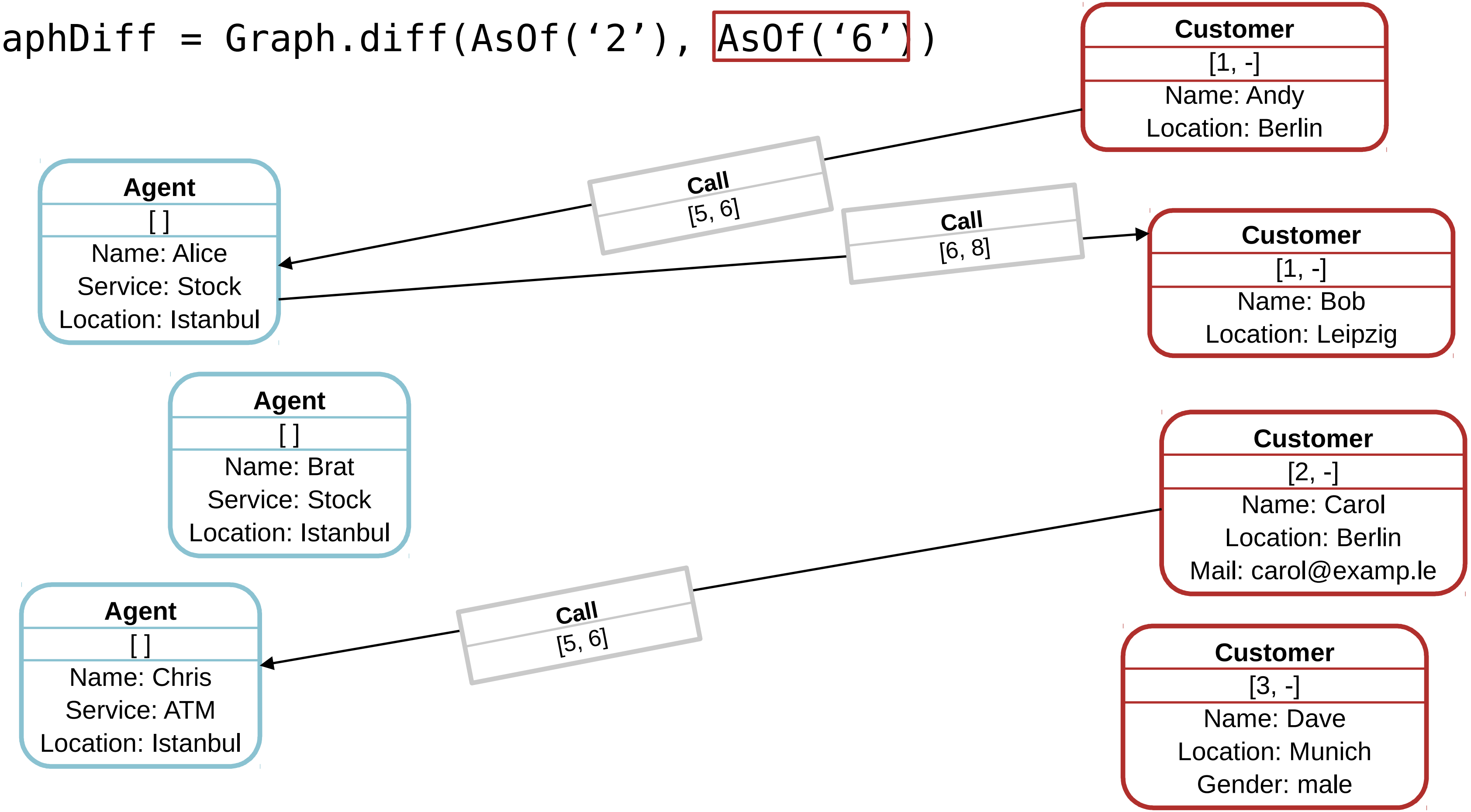
```
GraphDiff = Graph.diff(AsOf('2'), AsOf('6'))
```



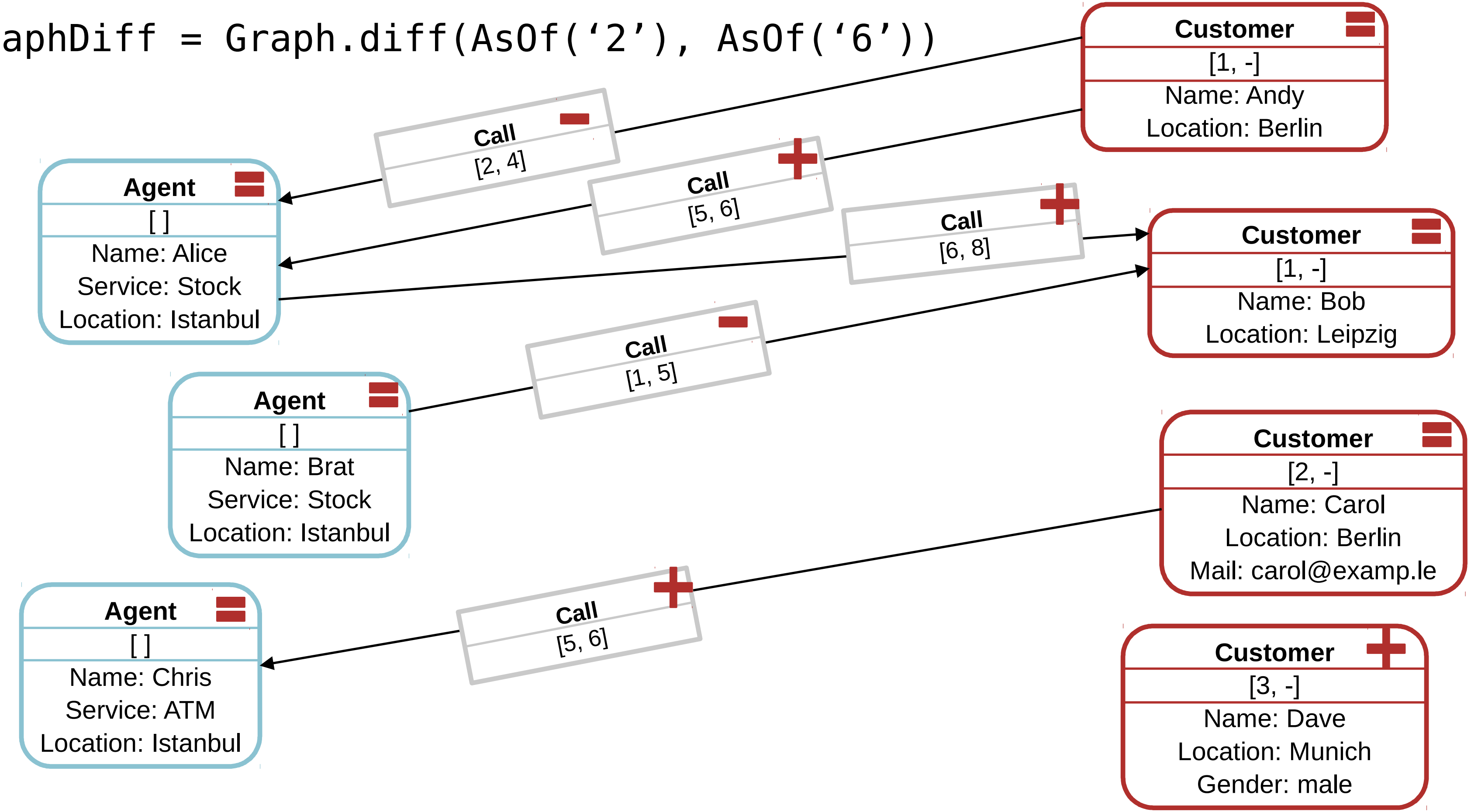
```
GraphDiff = Graph.diff(AsOf('2'), AsOf('6'))
```



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



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



OPERATORS

Transformation	Grouping
Snapshot	Graph Evolution

EPGM
TPGM

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?

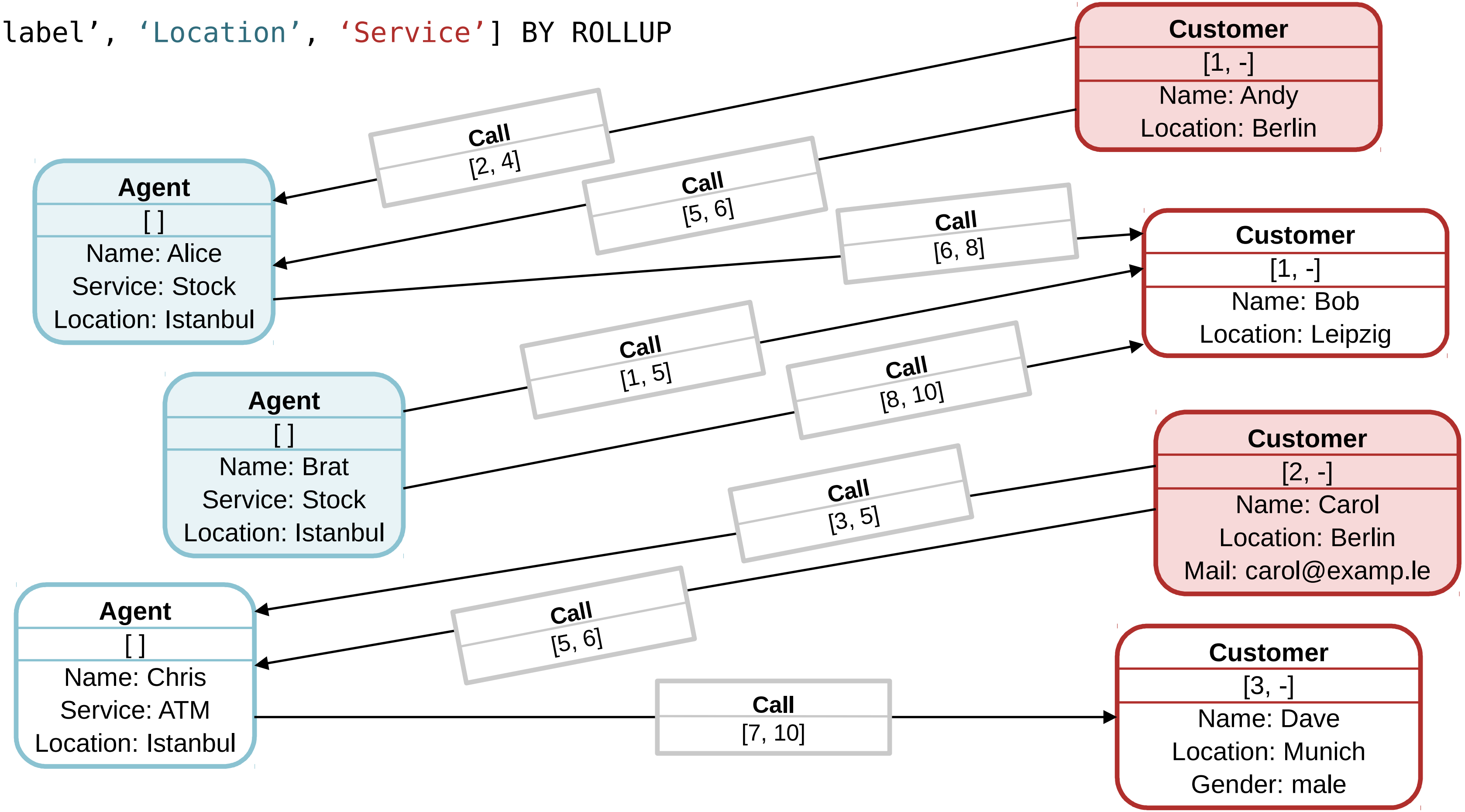
GROUPING

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

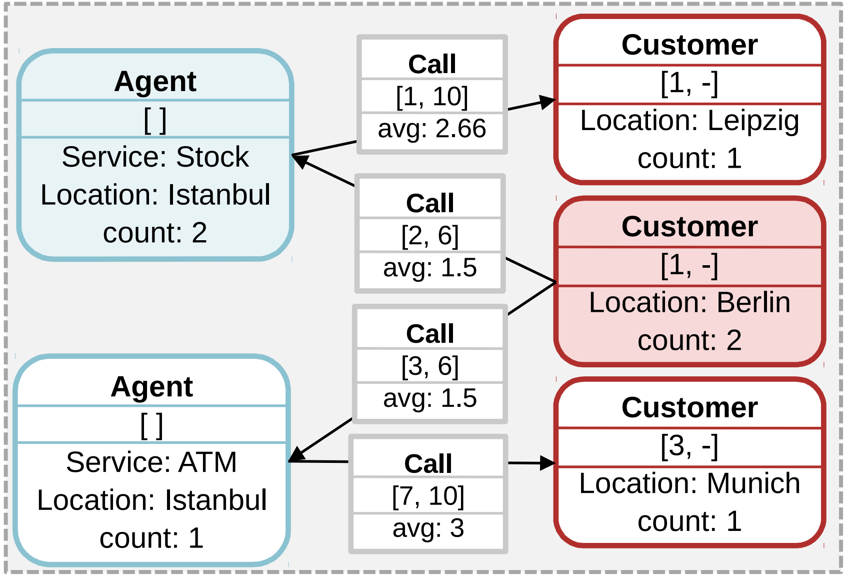
How long are customers talking with agents on average by location and service?

```
GraphCollection = Graph.groupBy(
    [‘:label’, ‘Location’, ‘Service’] BY ROLLUP,
    [superVertex[‘count’] = Count()],
    [‘:label’],
    [superEdge[‘avg’] = AvgDuration()]
)
```

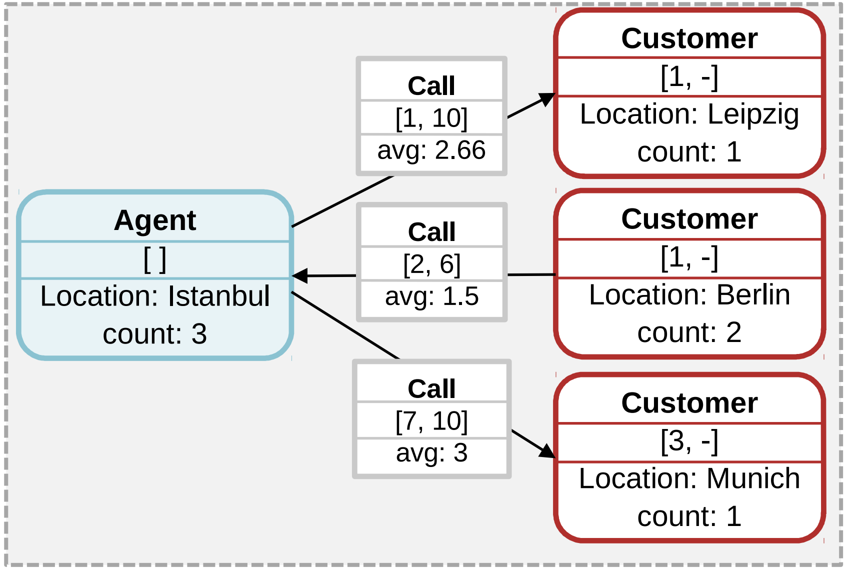
`[':label', 'Location', 'Service'] BY ROLLUP`



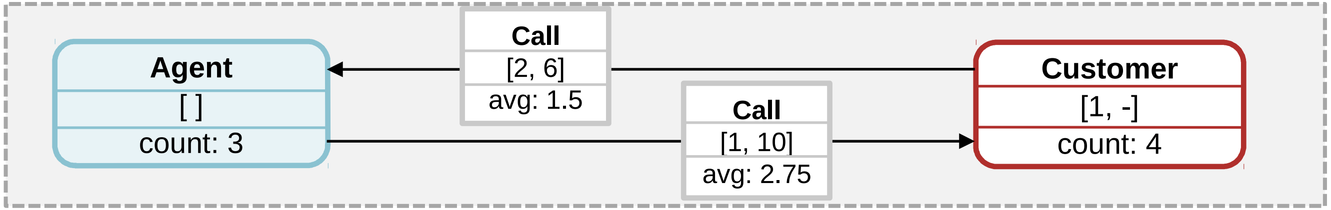
['`:label`', '`Location`', '`Service`'] BY ROLLUP



['`:label`', '`Location`', '`Service`'] BY ROLLUP



['`:label`', '`Location`', '`Service`'] BY ROLLUP



UNIVERSITÄT
LEIPZIG

Visit gradoop @ www.gradoop.com

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