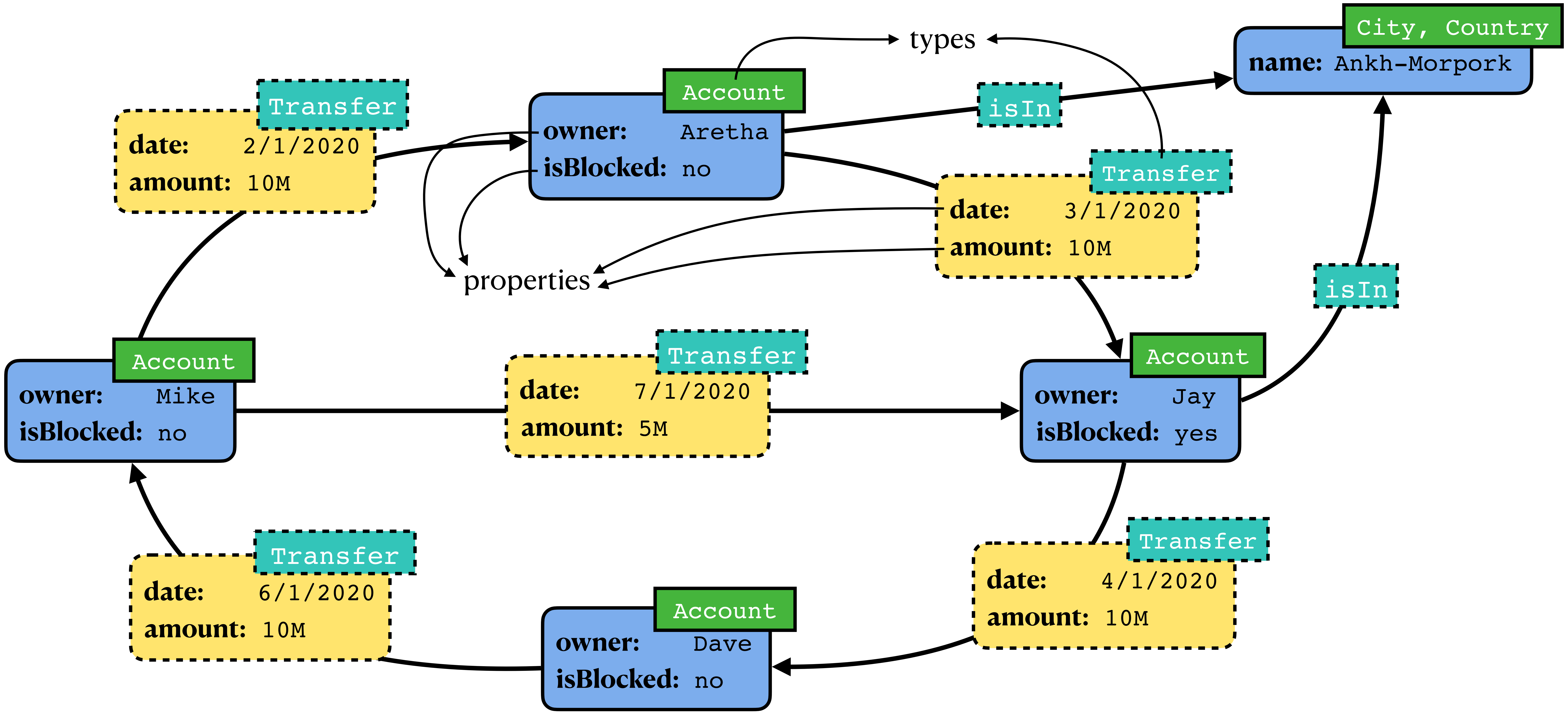
# Graph Pattern Matching in GQL and SQL/PGQ

SIGMOD 2022 (Industry Paper) + LDBC Presentation

# Property Graphs

- A data model based on graphs where both nodes and relationships can have properties (attributes) and types (label)



# Property Graphs in Industry

- Multiple vendors:
  - Neo4j
  - Oracle
  - TigerGraph
  - Amazon
  - SAP
  - Redis
  - DataStax, etc.
- Widespread: used by 75% of Fortune 100 companies
- Prediction (Gartner): in the next 5 years, up to **80% of all data analytics** tasks will involve graph databases
- Prediction (IDG): Graph database market will experience **600% growth** over the next decade

# Querying Property Graphs

- Multiple declarative languages: Cypher, PGQL, GSQL, G-Core, etc…
  - They look like dialects of the same language rather then different ones
- New Standard: **GQL (Graph Query Language)**
  - in development since 2019
- Another standardization project: **SQL/PGQ** (SQL Property Graph Querying):
  - graphs are defined as views over a relational schema
  - in development since 2017
- The engine of a graph query language: **graph pattern matching language (GPML)**
  - shared by GQL and SQL/PGQ

# Standards: Process

**ISO/IEC JTC1 SC32 WG3**

SQL Standard Committee:
9075-16 - SQL/PGQ
39075 - GQL Standard

**Linked Data Benchmark Council**

Liaison

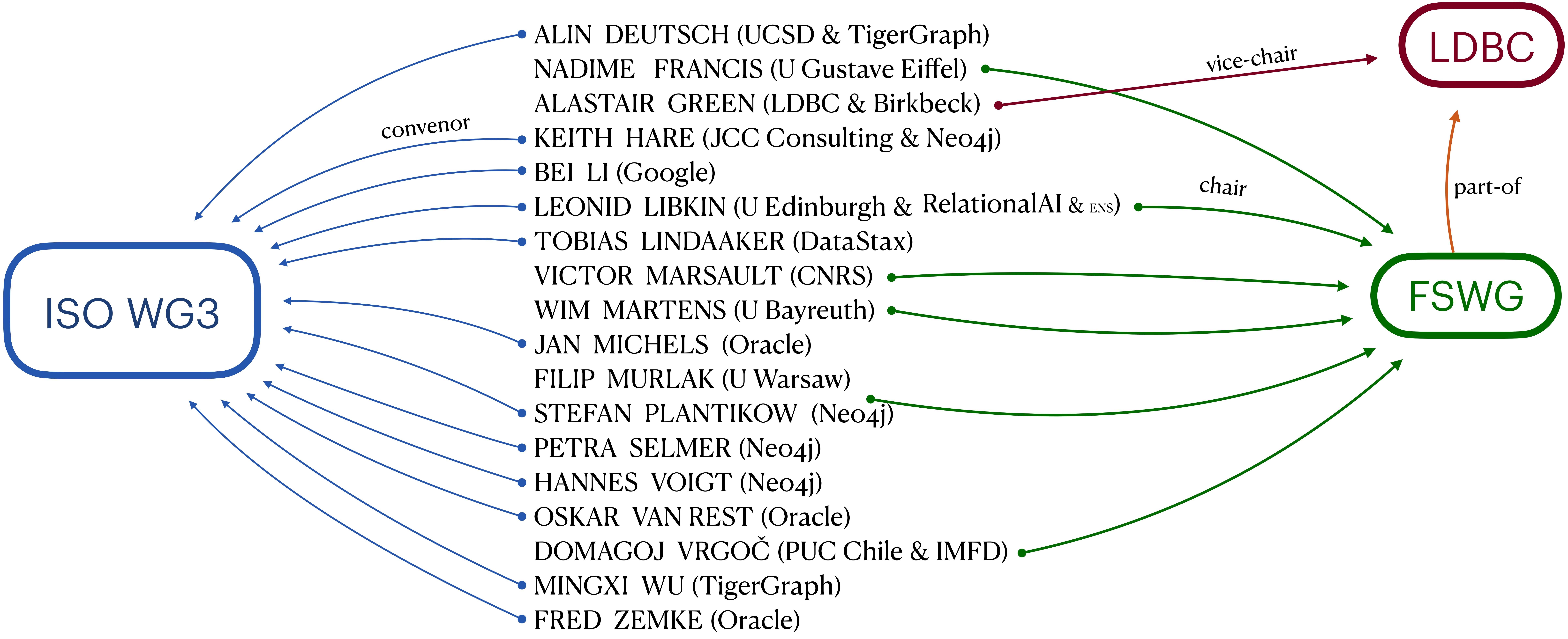
LDBC

Working groups

Formal Semantics Working Group (FSWG)

analyzes design decisions;
provides feedback

# Authors

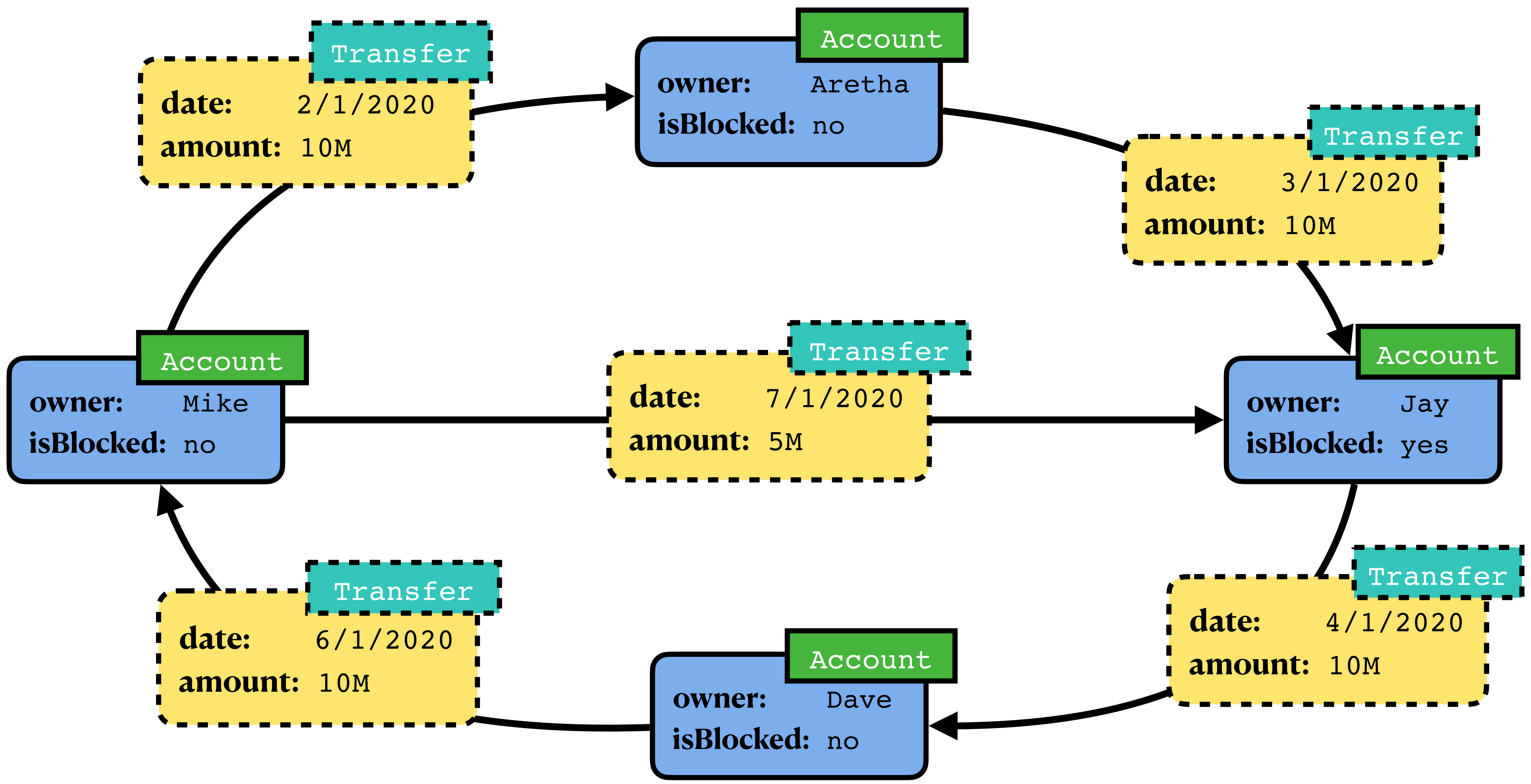


ISO WG3
convenor
ALIN DEUTSCH (UCSD & TigerGraph)
NADIME FRANCIS (U Gustave Eiffel)
ALASTAIR GREEN (LDBC & Birkbeck)
KEITH HARE (JCC Consulting & Neo4j)
BEI LI (Google)
LEONID LIBKIN (U Edinburgh & RelationalAI & ENS)
TOBIAS LINDAAKER (DataStax)
VICTOR MARSAULT (CNRS)
WIM MARTENS (U Bayreuth)
JAN MICHELS (Oracle)
FILIP MURLAK (U Warsaw)
STEFAN PLANTIKOW (Neo4j)
PETRA SELMER (Neo4j)
HANNES VOIGT (Neo4j)
OSKAR VAN REST (Oracle)
DOMAGOJ VRGOČ (PUC Chile & IMFD)
MINGXI WU (TigerGraph)
FRED ZEMKE (Oracle)
vice-chair
LDBC
chair
part-of
FSWG

# GPML: Nodes

Selecting nodes:

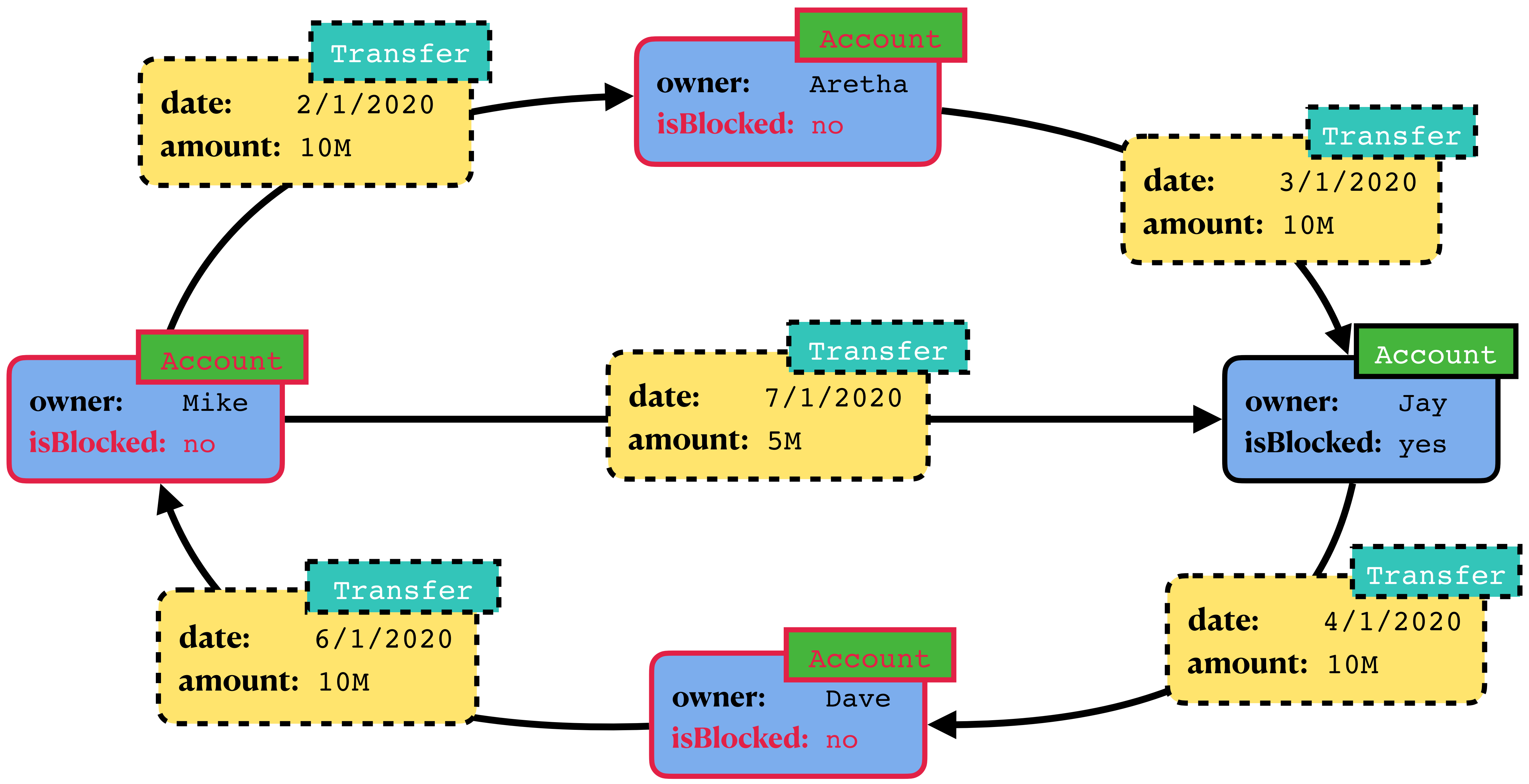
```
MATCH (x:Account)
WHERE x.isBlocked = 'no'
```



# GPML: Nodes

Selecting nodes:

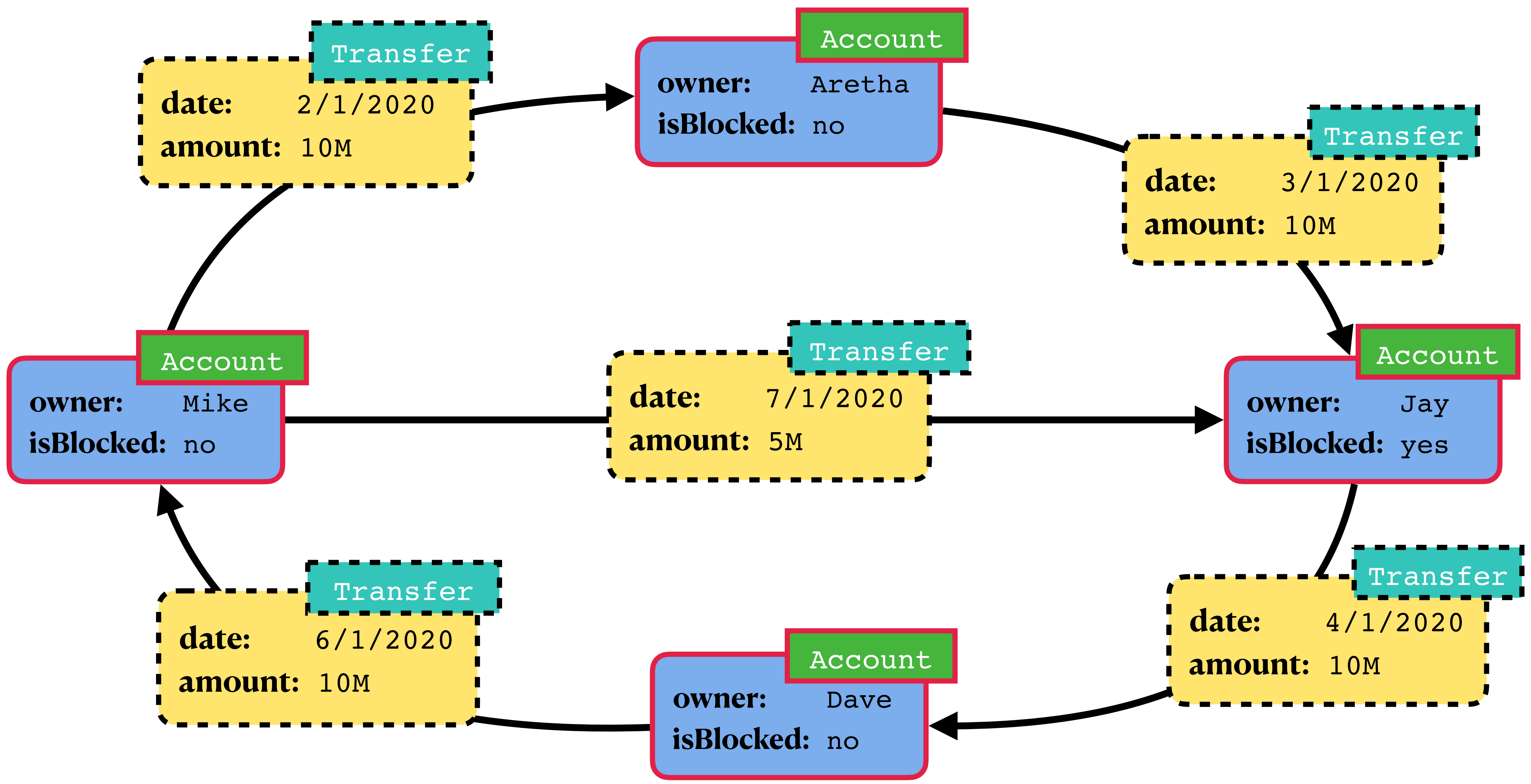
```
MATCH (x:Account)
WHERE x.isBlocked = 'no'
```



# GPML: Nodes

All nodes:

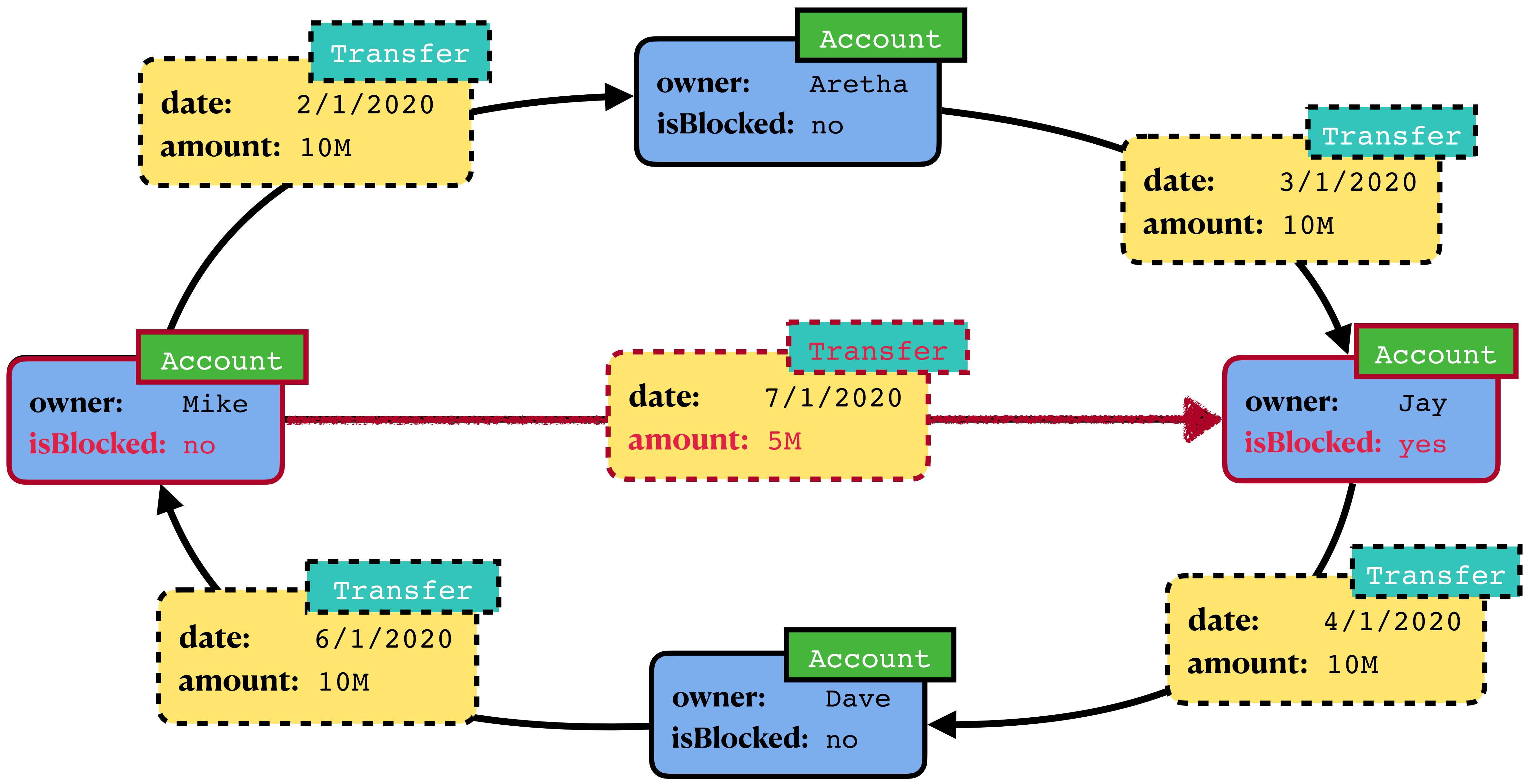
```
MATCH (x)
```



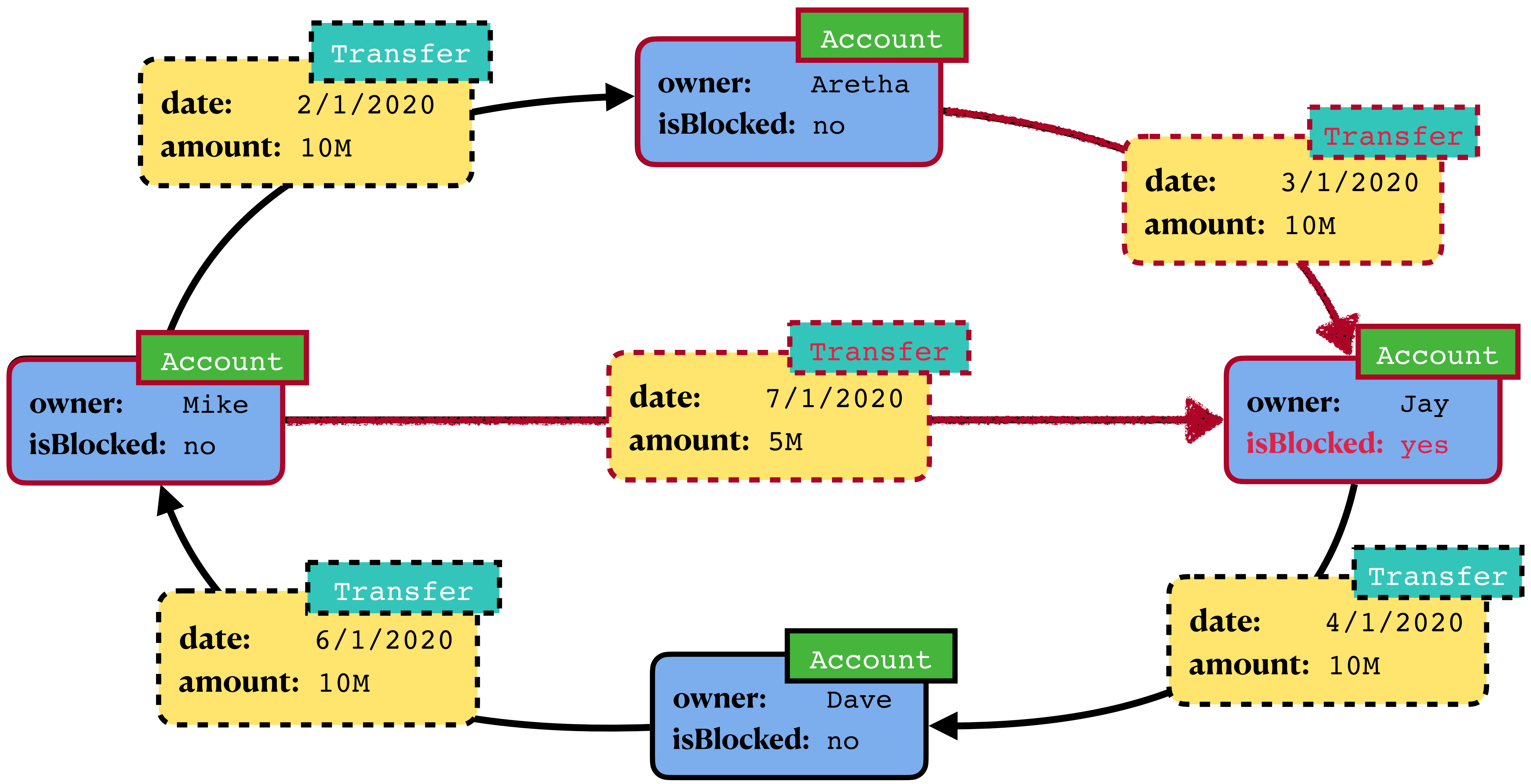
# GPML: Paths

Combining nodes and edges:

```
MATCH (x)-[e1:Transfer]->(y)
WHERE x.isBlocked = 'no'
  AND y.isBlocked = 'yes'
  AND e1.amount <= 5M
```



# GPML: Paths

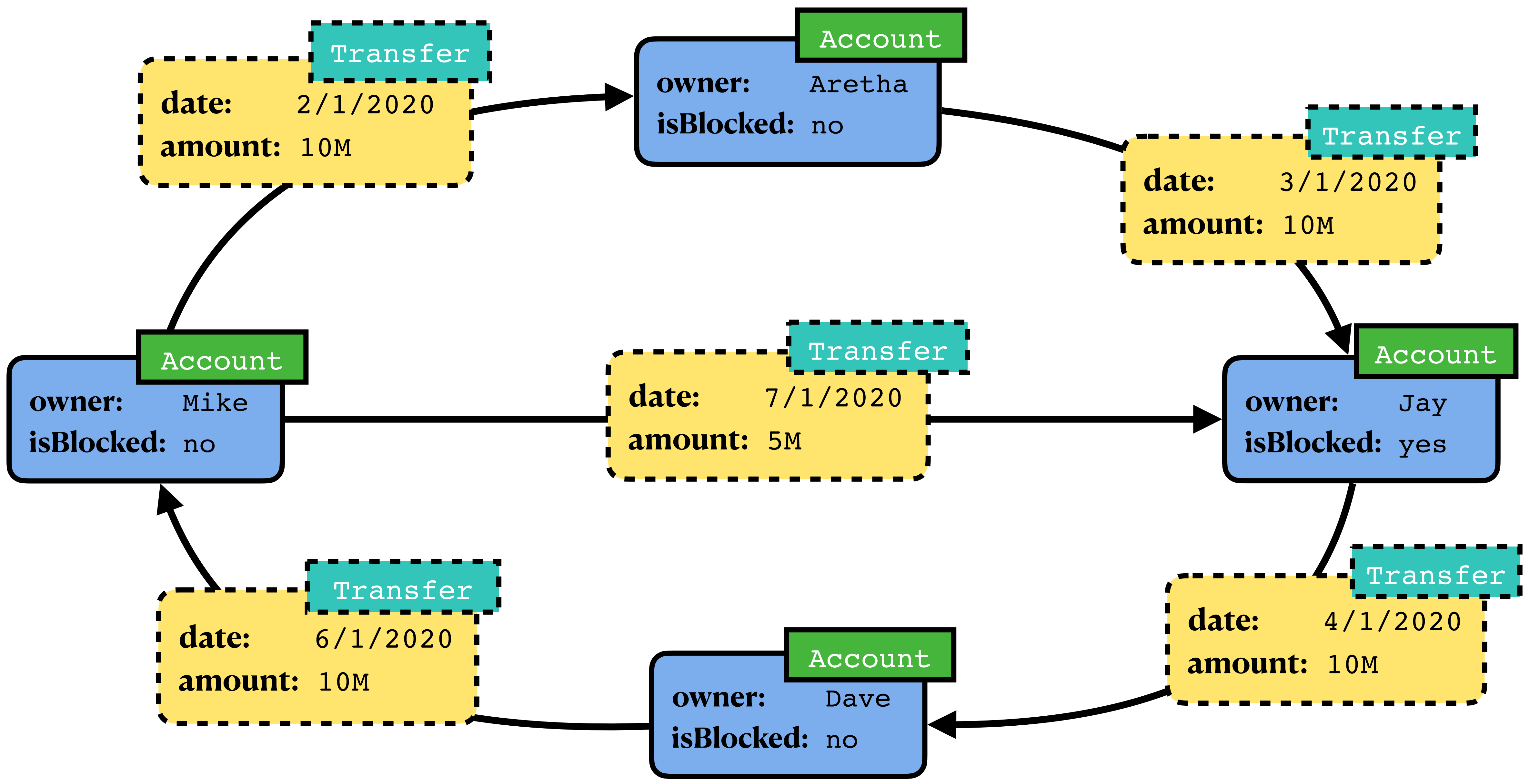



Multiple edge options: ~, -, ->, <-

Longer paths are defined via ASCII art:

```
MATCH (x)-[:Transfer]->(y)<-[:Transfer]-(z)
WHERE y.isBlocked = 'yes'
```

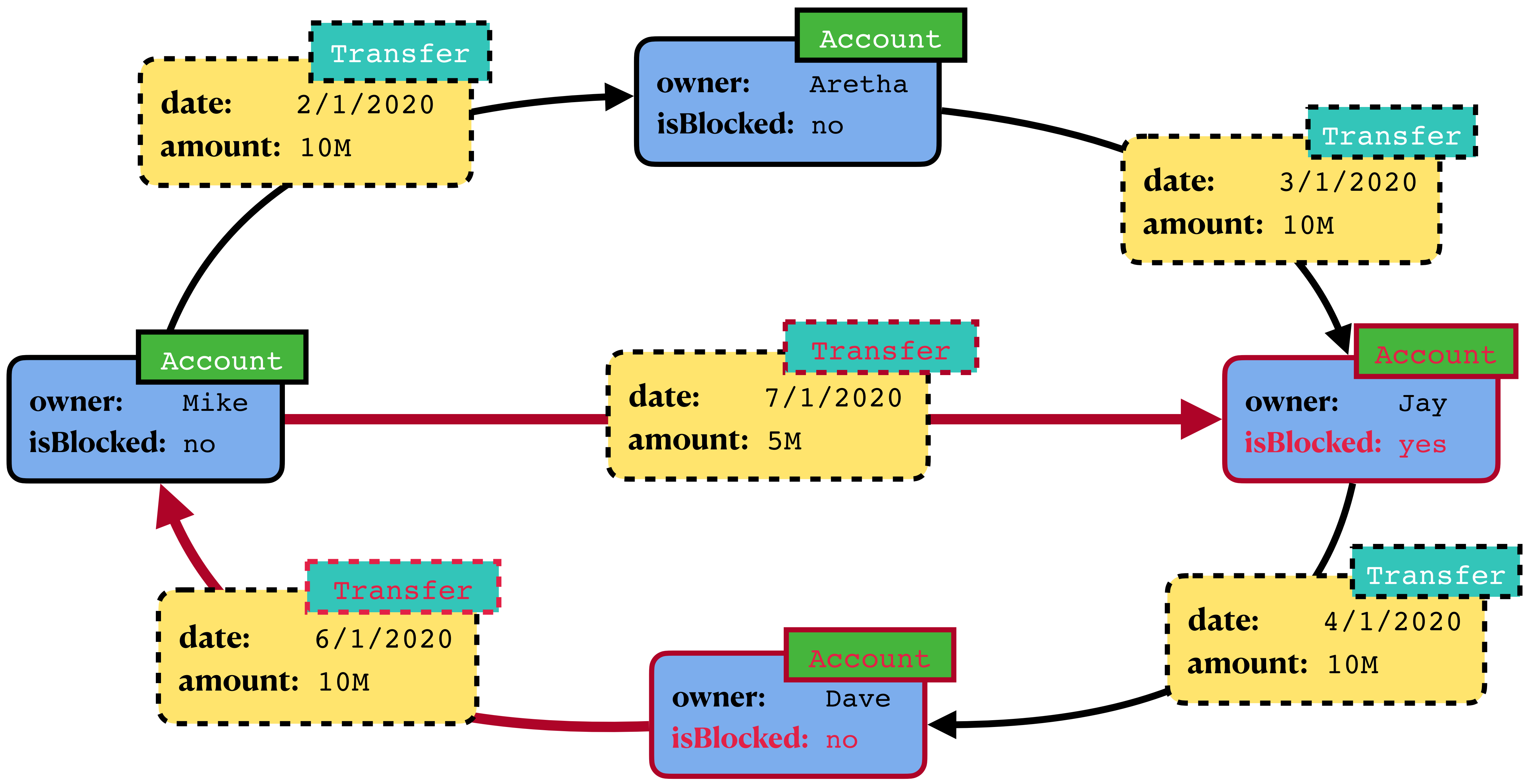
# GPML: Path traversal




Specifying graph traversal:

```
MATCH (x:Account)-[t:Transfer]->{2,5}(y:Account)
WHERE x.isBlocked = 'no' AND y.isBlocked = 'yes'
```

# GPML: Path traversal

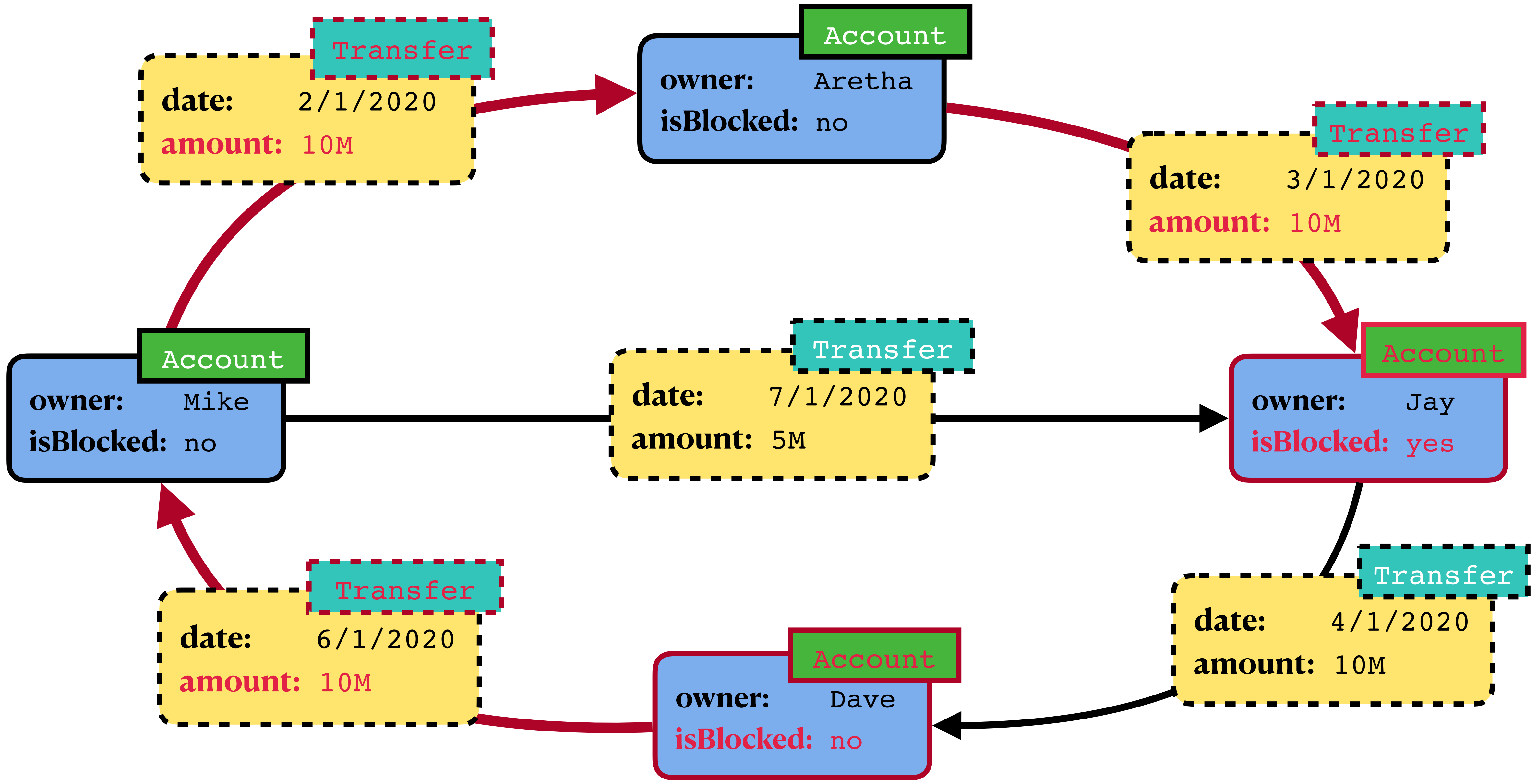



Specifying graph traversal:

```
MATCH (x:Account)-[t:Transfer]->{2,5}(y:Account)
WHERE x.isBlocked = 'no' AND y.isBlocked = 'yes'
```

# GPML: Path traversal

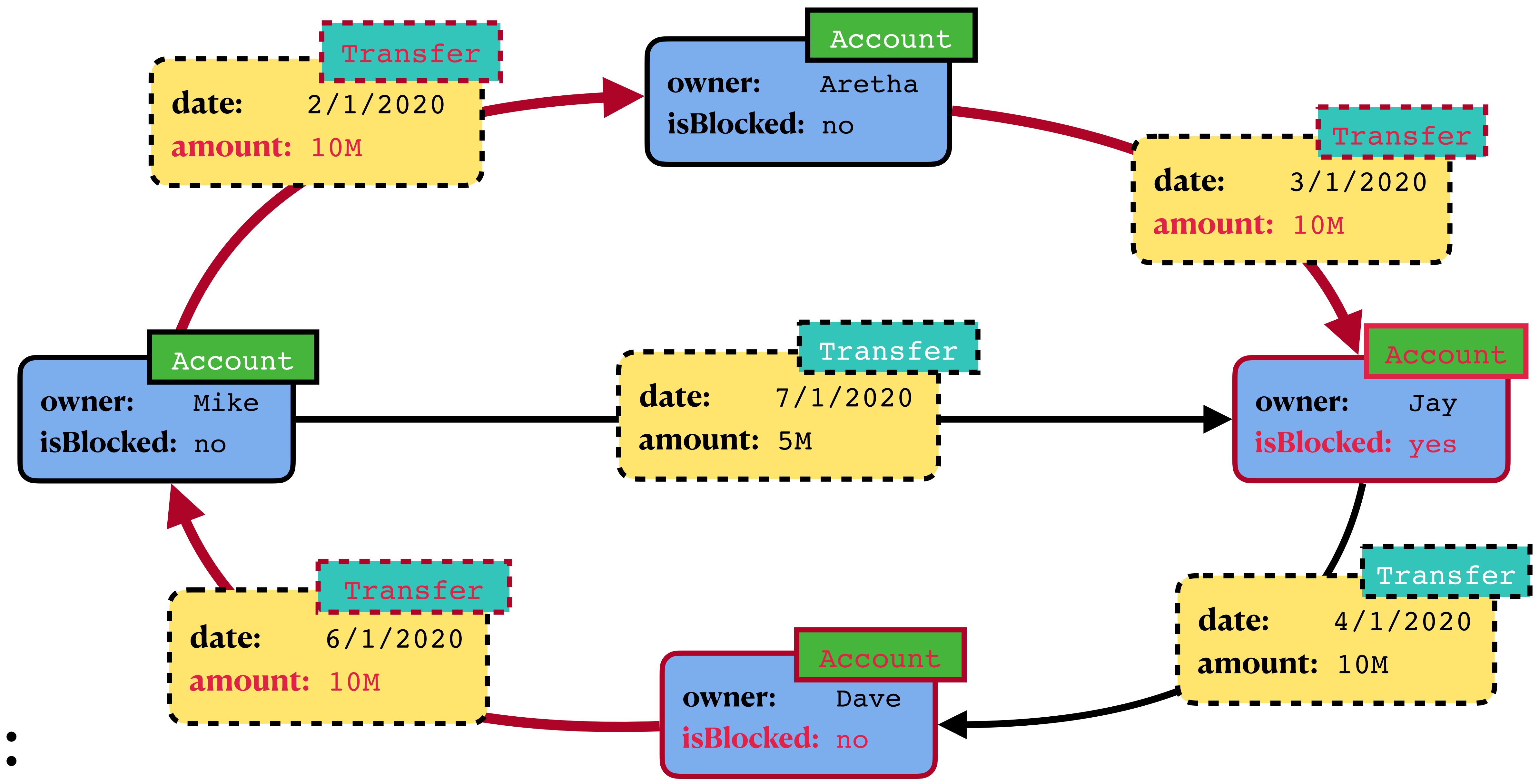
Allowed quantifiers: `{m,n}`, `*`, `+`




Path conditions can be added:

```
MATCH (x:Account)
      -[t:Transfer WHERE t.amount > 7M]->{2,5}
      (y:Account)
WHERE x.isBlocked = 'no' AND y.isBlocked = 'yes'
```

# GPML: Path traversal



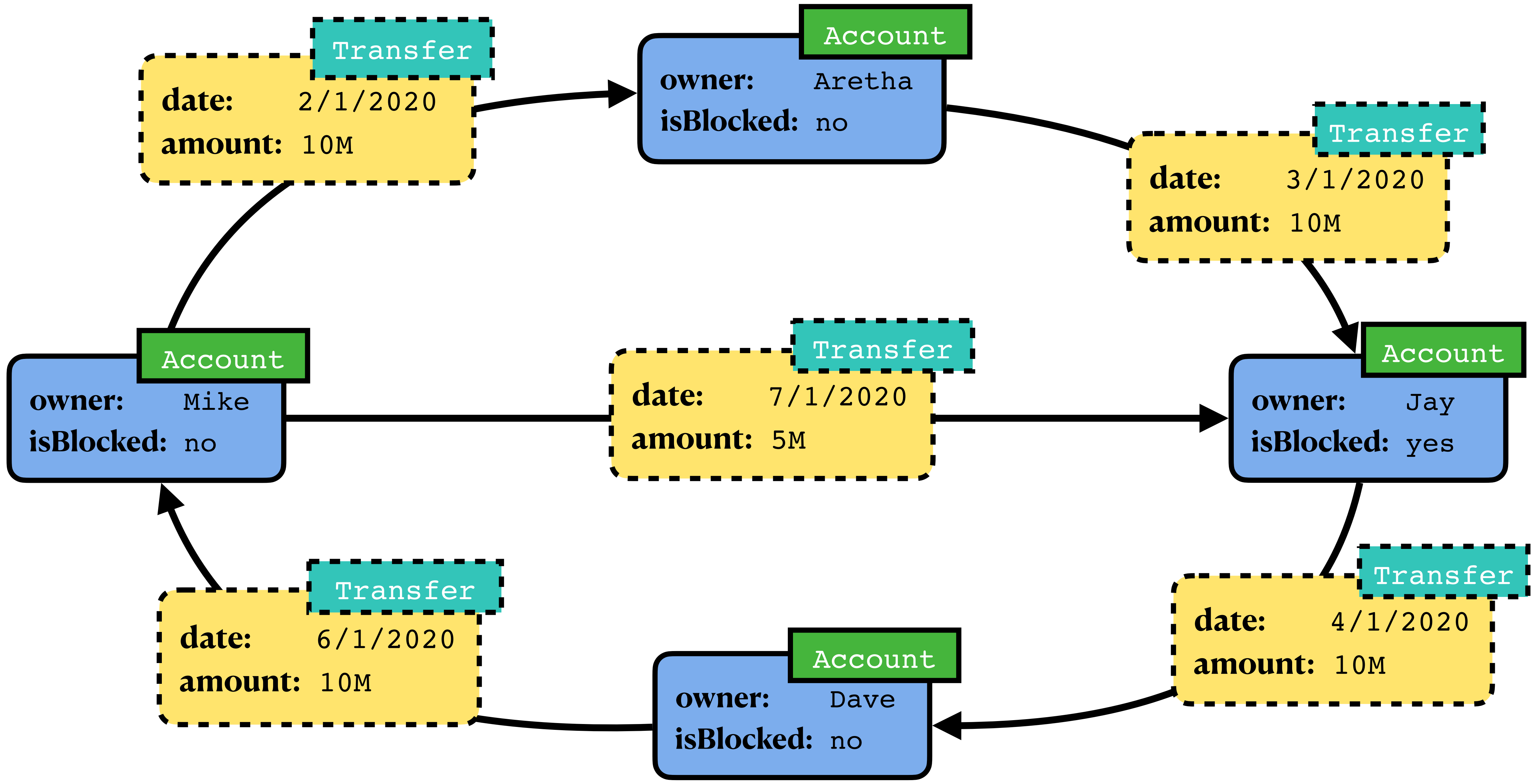

It's not just single edges that can be repeated:

```
MATCH (x:Account)
      [ -[t:Transfer WHERE t.amount > 7M]->
        [t1: Transfer WHERE t1.amount > 3M] -> ]{2,5}
      (y:Account)
WHERE x.isBlocked = 'no' AND y.isBlocked = 'yes'
```

# GPML: Simple, Trail, Shortest

Issues when returning paths:

```
MATCH
    p = (x WHERE x.owner = 'Mike')
            -[:Transfer]->*
        (y WHERE y.owner = 'Jay')
```



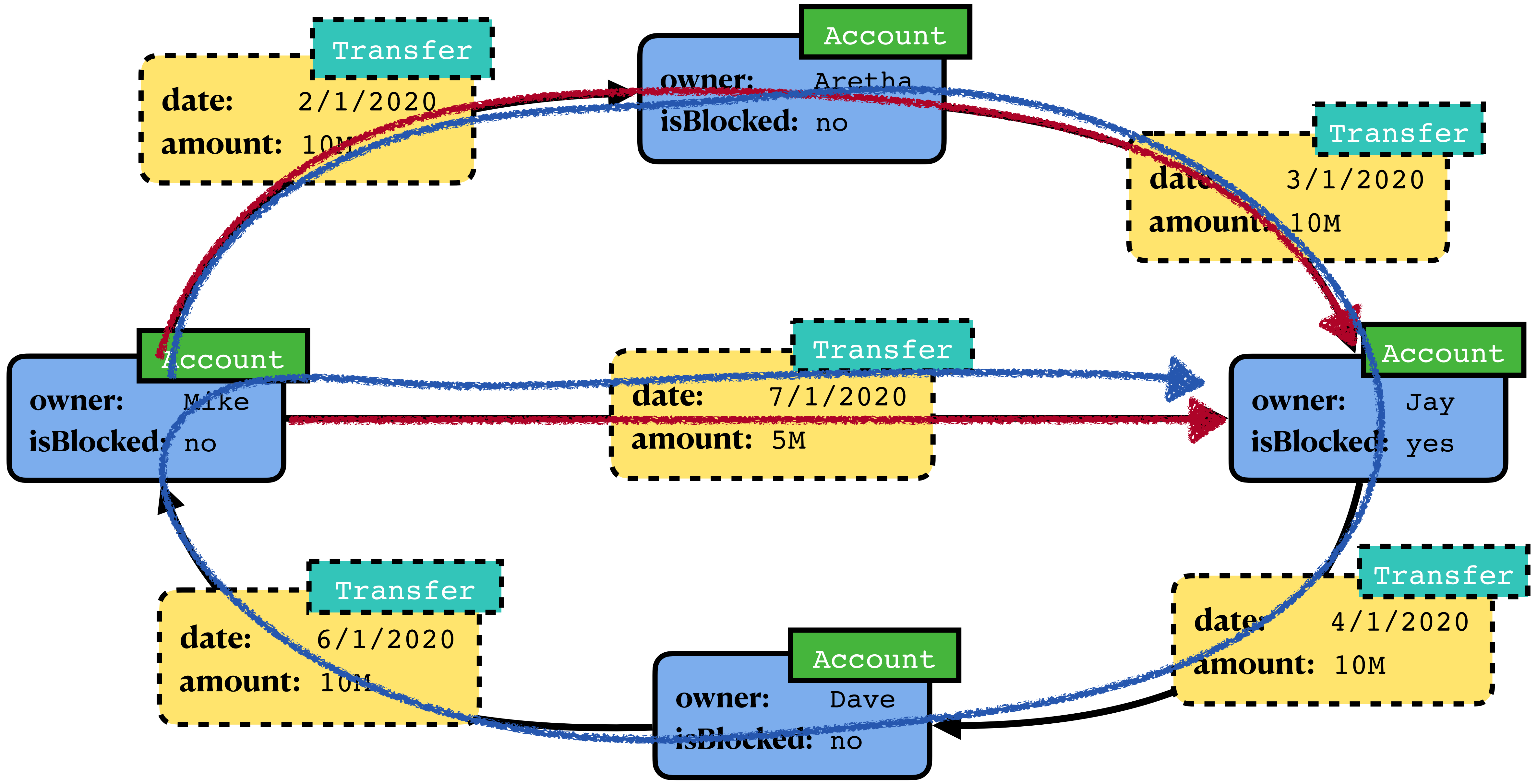

To deal with this GPML allows restrictors and selectors:

- **restrictors** restrict the set of considered paths to be finite;
- **selectors** filter out the results to assure finiteness.

# GPML: Simple, Trail, Shortest

How do **restrictors** work?

```
MATCH SIMPLE
    p = (x WHERE x.owner = 'Mike')
              -[:Transfer]->*
        (y WHERE y.owner = 'Jay')
```

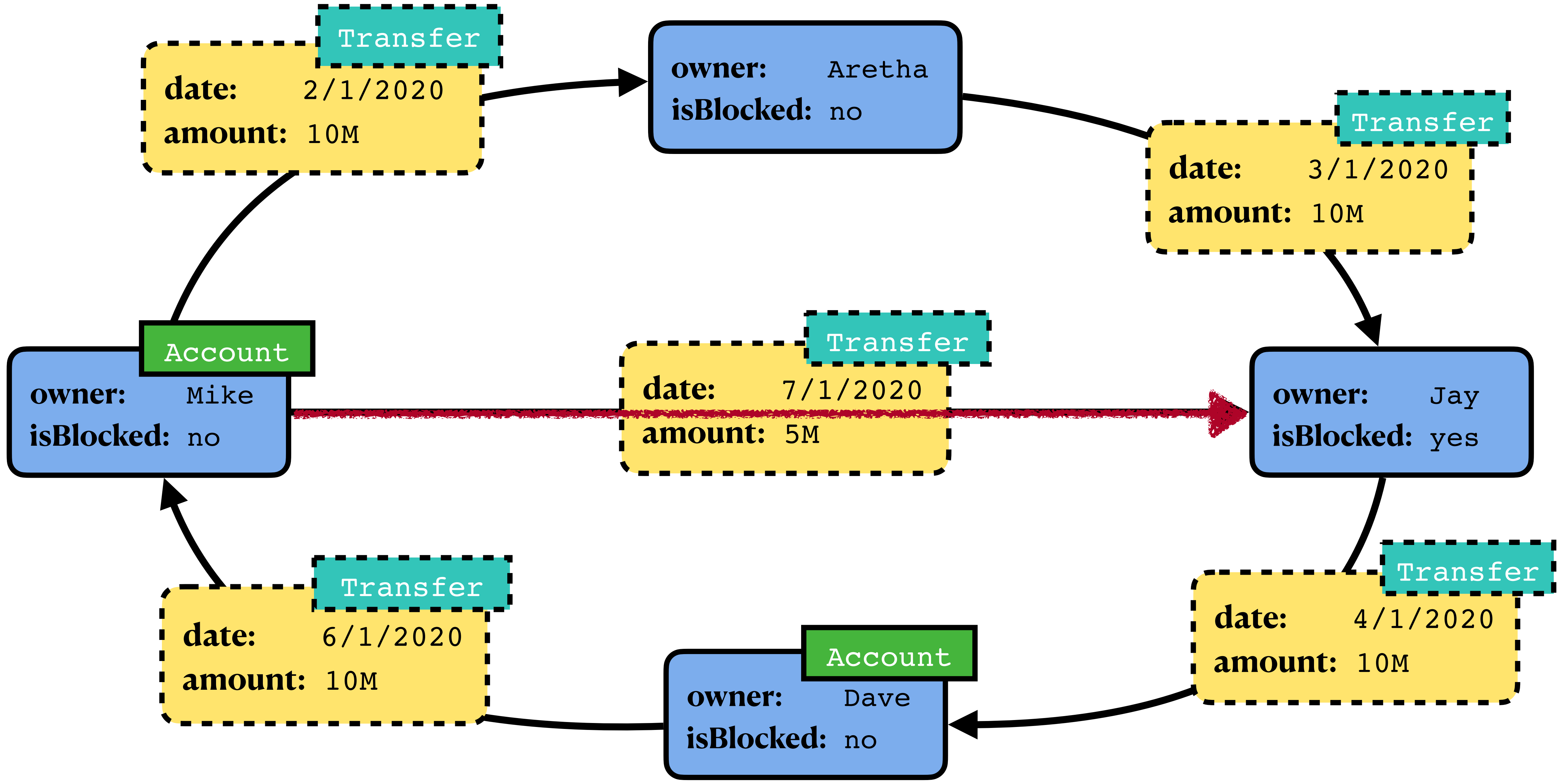



Also available: **`TRAIL`**, **`ACYCLIC`**

# GPML: Simple, Trail, Shortest

How **selectors** work?

```
MATCH ALL SHORTEST
    p = (x WHERE x.owner = 'Mike')
            -[:Transfer]->*
        (y WHERE y.owner = 'Jay')
```




Also available: **`ANY SHORTEST`**

Can be combined with restrictors

# GPML: Union, Optional

Two types of union: set-based and multiset-based (SQL `UNION` vs `UNION ALL`)

Conditional matches:

```
MATCH (x)-[:Transfer]->(y)[-[:Transfer]->(z)]?
WHERE y.isBlocked = 'yes'
```

*Transfers to a blocked account, and, if available, all outgoing transfers.*

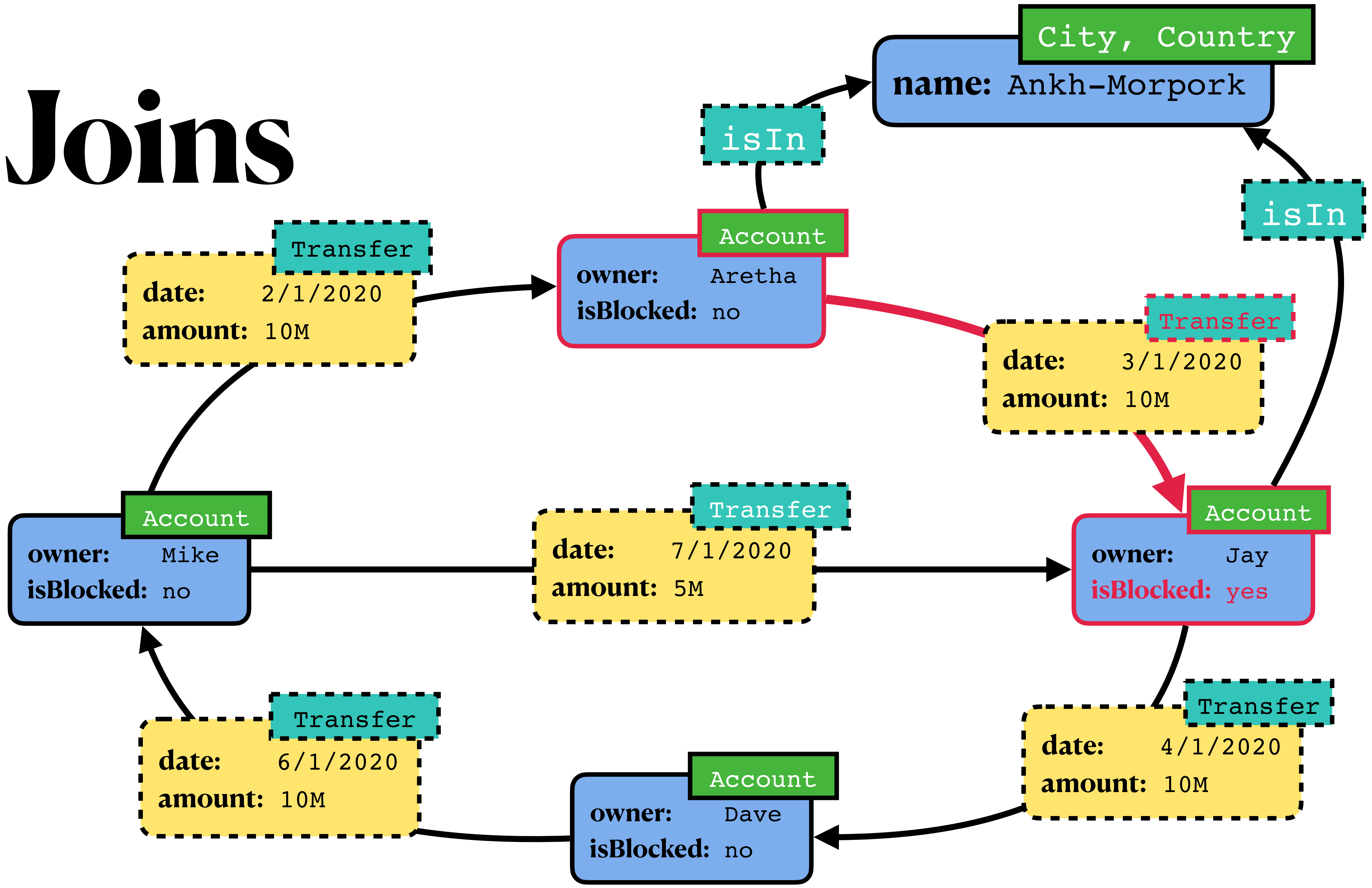
# GPML: Joins

Finally, we can combine all these into a single query:

```
MATCH TRAIL p = (x) -[:Transfer]->  (y),
                (y) -[:Transfer]->+ (x),
                (x:Account)-[:isIn]->(c1:City),
                (y:Account)-[:isIn]->(c2:City)
WHERE c1.name = c2.name AND y.isBlocked = 'yes'
```

*Accounts in the same city, with both a direct transfer between them, and also a path that links them in the other direction (i.e. Aretha is laundering money).*

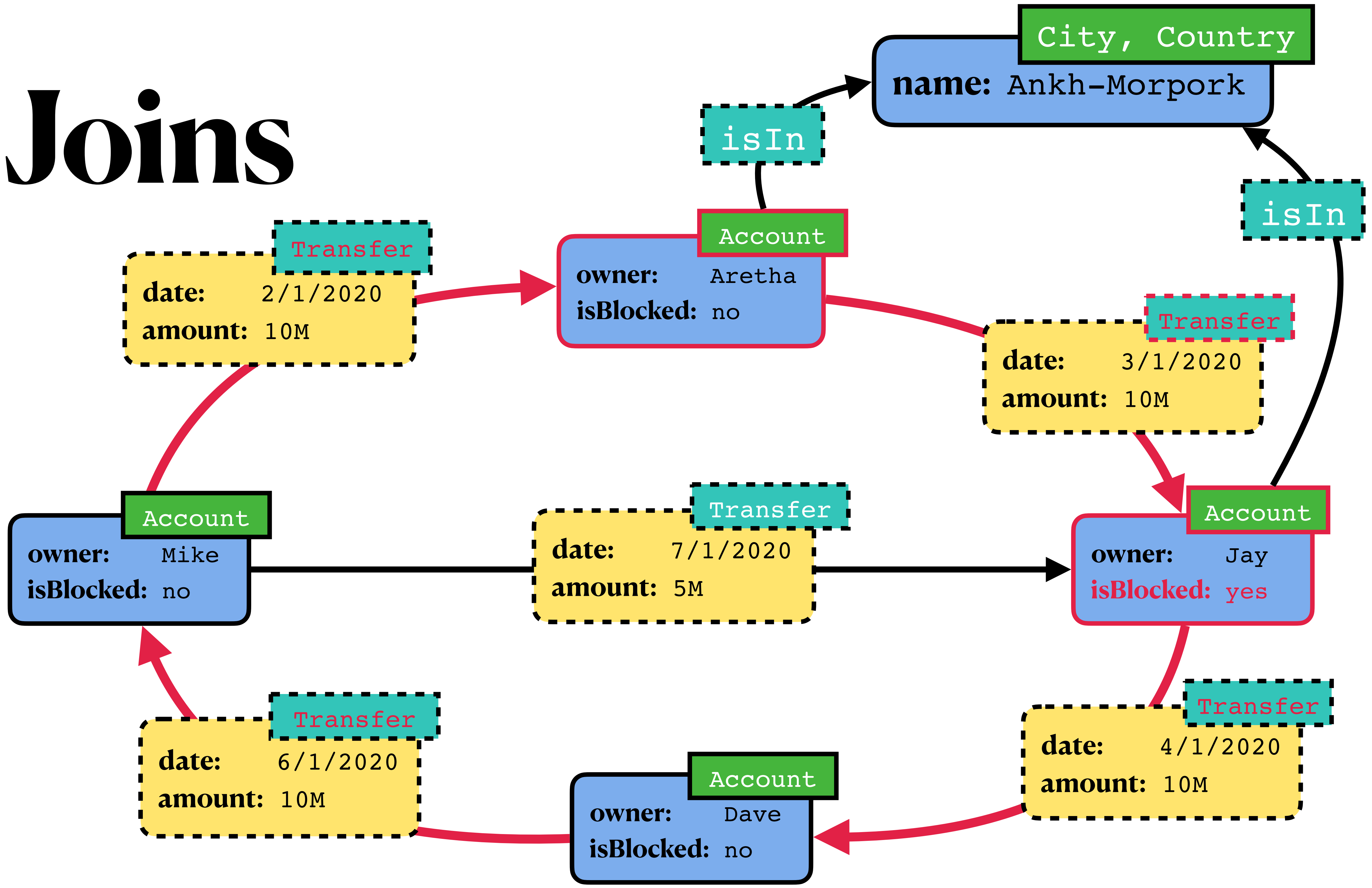
# GPML: Joins




Money laundering scheme:

```
MATCH TRAIL p = (x) -[:Transfer]->  (y),
                (y) -[:Transfer]->+ (x),
                (x:Account)-[:isIn]->(c1:City),
                (y:Account)-[:isIn]->(c2:City)
WHERE c1.name = c2.name AND y.isBlocked = 'yes'
```

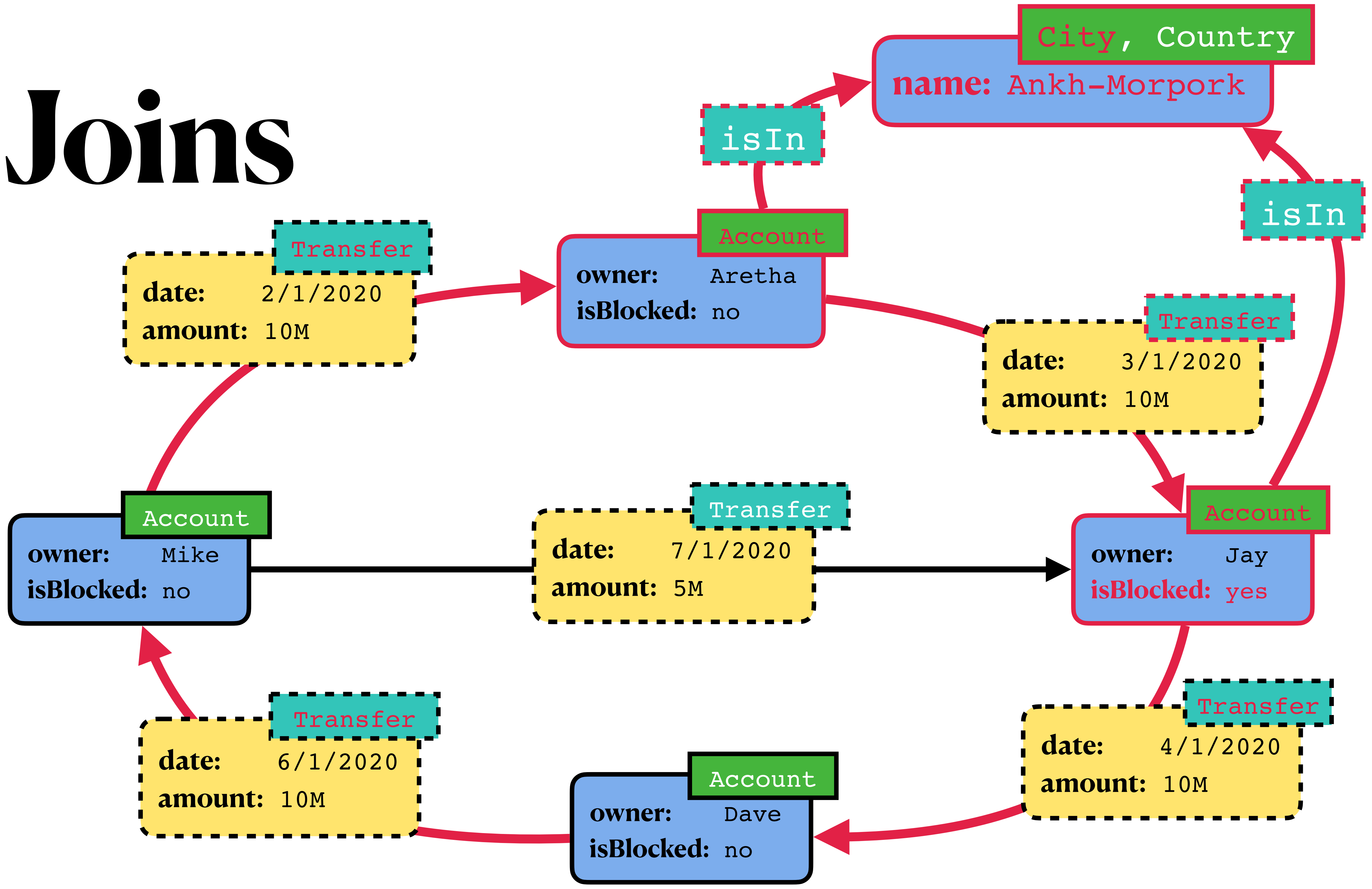
# GPML: Joins




Money laundering scheme:

```
MATCH TRAIL p = (x) -[:Transfer]->  (y),
                (y) -[:Transfer]->+ (x),
                (x:Account)-[:isIn]->(c1:City),
                (y:Account)-[:isIn]->(c2:City)
WHERE c1.name = c2.name AND y.isBlocked = 'yes'
```

# GPML: Joins



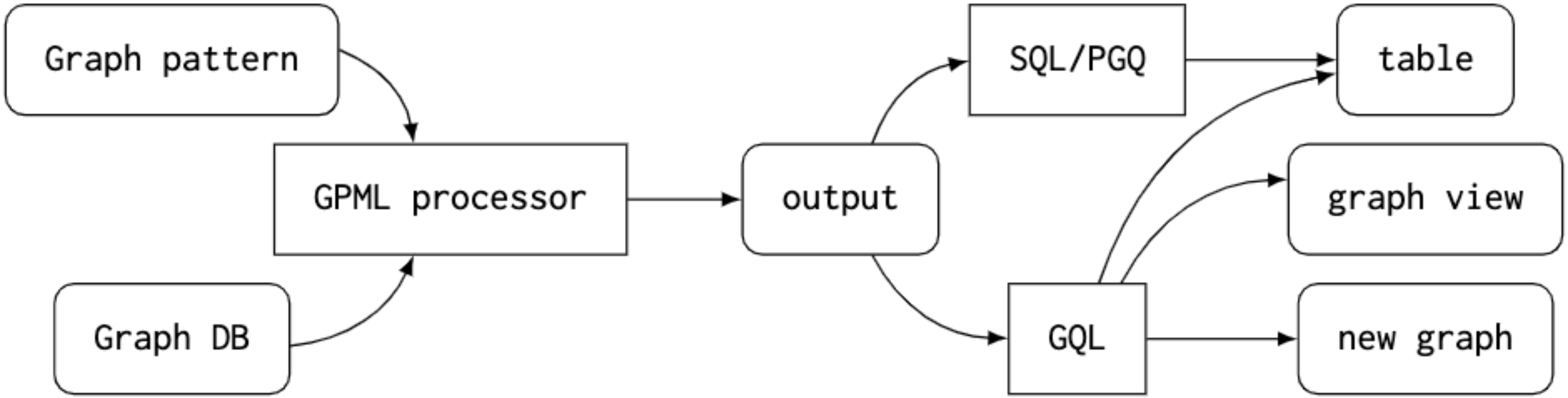

Money laundering scheme:

```
MATCH TRAIL p = (x) -[:Transfer]->  (y),
                (y) -[:Transfer]->+ (x),
                (x:Account)-[:isIn]->(c1:City),
                (y:Account)-[:isIn]->(c2:City)
WHERE c1.name = c2.name AND y.isBlocked = 'yes'
```

# GPML: Output

GPML output: a data structure that combines paths in graphs with bindings of variables

Can be embedded in GQL and in SQL/PGQ.



# Timeline to Standards

| Date | SQL/PGQ | GQL |
|---|---|---|
| 2017 | Work started | |
| 2018 | | Work started |
| 2021-02-07 | CD Ballot End | |
| 2022-02-20 | | CD Ballot End |
| 2022-12-04 | DIS Ballot End | |
| 2023-01-30 | Final Text to ISO | |
| 2023-03-13 | **SQL/PGQ IS Published** | |
| 2023-05-21 | | DIS Ballot End |
| 2023-07-30 | | Final Text to ISO |
| 2023-09-10 | | **GQL IS Published** |

# Research Challenges

- Find a workable abstraction of GPML for research (systems and theory).
- Support aggregation

```
MATCH (x)-[e:Flight]->*(y)
WHERE x.name='Zembla'
  AND y.name='Ankh-Morpork'
  AND SUM(e.duration) < 24
```

- Optimize GMPL processing (vendors already working on it).