Property Graph Query Language
Composition & Co.
(with Corrections)

Romans Kasperovics, SAP HANA Graph
Fifth openCypher Implementers Meeting
March 6, 2019 Berlin (Germany)
Fixed Subgraph Query (and very boring one)

```
MATCH (U:USER)
RETURN U.NAME, U.EMAIL
```

```
SELECT U.NAME, U.EMAIL
FROM USER U
```
Fixed Subgraph Query with Join

```
MATCH (U:USER), (P:POST)
WHERE P.CREATOR = U.NAME
RETURN U.NAME, U.EMAIL, P.TITLE
```

```
SELECT U.NAME, U.EMAIL, P.TITLE
FROM USER U, POST P
WHERE P.CREATOR = U.NAME
```
Subqueries

```sql
SELECT U.NAME, U.EMAIL, P.TITLE
FROM (SELECT NAME
     FROM USER
     ORDER BY REG_DATE DESC
     LIMIT 5) U, POST P
WHERE P.CREATOR = U.NAME
```

```cypher
MATCH (U:MATCH({_U:USER})
RETURN _U.NAME
ORDER BY _U.REG_DATE DESC
LIMIT 5), (P:POST)
WHERE P.CREATOR = U.NAME
RETURN U.NAME, U.EMAIL, P.TITLE
```
Subqueries (with Valid Cypher + SQL-PG Syntax)

```
SELECT U.NAME, U.EMAIL, P.TITLE
FROM (SELECT NAME
      FROM USER
      ORDER BY REG_DATE DESC
      LIMIT 5) U, POST P
WHERE P.CREATOR = U.NAME

MATCH (U:USER)
WITH U
ORDER BY U.REG_DATE DESC LIMIT 5
MATCH (P:POST)
WHERE P.CREATOR = U.NAME
RETURN U.NAME, U.EMAIL, P.TITLE
```

```
SELECT U.NAME, U.EMAIL, P.TITLE
FROM (SELECT NAME
      FROM USER
      ORDER BY REG_DATE DESC
      LIMIT 5) U
JOIN GRAPH_TABLE(
    MATCH (P IS POST)
    WHERE P.CREATOR = U.NAME
    COLUMNS (P.TITLE)
  ) P
```
Constructing Graph View on Tables using GRAPH_QUERY

Current SQL-PG proposal:

CREATE PROPERTY GRAPH "forum"."graph"
    VERTEX TABLES ("forum"."POST" LABEL "post")
    EDGE TABLES ("forum"."REFERENCE"
        SOURCE "forum"."post" DESTINATION "forum"."post" LABEL "refersTo")

Using GRAPH_QUERY with embedded SQL:

CREATE GRAPH VIEW "forum"."graph" AS GRAPH_QUERY(
    MATCH (A AS (SELECT * FROM "forum"."POST"))
    OPTIONAL (A)-[E AS "forum"."REFERENCE"]->(B AS "forum"."POST")
    CONSTRUCT (A IS "post")-[E IS "refersTo"]->(B IS "post")
);

✓ Has "view" in its name
✓ Same language for queries & view definitions
✓ Enables graph views on tables and on other graphs
✗ No incremental construction
✗ No empty properties
Constructing Graph View on Tables using GRAPH_OBJECT

Current SQL-PG proposal:

```
CREATE PROPERTY GRAPH "forum"."graph"
   VERTEX TABLES ("forum"."POST" LABEL "post")
   EDGE TABLES ("forum"."REFERENCE" SOURCE "forum"."post"
                DESTINATION "forum"."post" LABEL "refersTo")
);
```

Using GRAPH_OBJECT:

```
CREATE GRAPH VIEW "forum"."graph" AS
 GRAPH_OBJECT(   VERTEX TABLES ("forum"."POST" LABEL "post")
   EDGE TABLES ("forum"."REFERENCE" SOURCE "forum"."post"
                DESTINATION "forum"."post" LABEL "refersTo")
);
```

✓ Has "view" in its name
✗ No incremental construction
Fixed Subgraph Reachability Join

\[
\begin{align*}
\text{MATCH} & \quad (U:USER), \ (P:POST) \\
\text{WHERE} & \quad (U)-[*]-(P) \\
\text{RETURN} & \quad U.NAME, \ U.EMAIL, \ P.TITLE \\
\end{align*}
\]
Variable-Length Results $\rightarrow$ JSON

MATCH S=SHORTEST((U:USER)-[*1..1000]-(P:POST))
RETURN U.NAME, P.TITLE, S

<table>
<thead>
<tr>
<th>NAME</th>
<th>TITLE</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>John1973</td>
<td>Graph query languages</td>
<td><img src="image" alt="JSON result" /></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[[&quot;NAME&quot;: &quot;John1973&quot;, &quot;EMAIL&quot;: &quot;<a href="mailto:john.doe1973@gmail.com">john.doe1973@gmail.com</a>&quot;, &quot;REG_DATE&quot;: &quot;2019-01-11 15:43:21&quot;, &quot;LABELS&quot;: [&quot;USER&quot;], &quot;LABELS&quot;: [&quot;createdBy&quot;], &quot;TITLE&quot;: &quot;Opinion on some irrelevant topic&quot;, &quot;CREATOR&quot;: &quot;John1973&quot;, &quot;CREATE_DATE&quot;: &quot;2019-03-04 12:00:01&quot;, &quot;CONTENT&quot;: &quot;Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.&quot;], [&quot;LABELS&quot;: [&quot;refersTo&quot;]], ...]]</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- JSON for variable-length results
  - Web-friendly ✓
  - Flexible ✓
  - Not selective ❌
  - Non-standardized schema ❌
  - Different from SQL types ❌

Standardization opportunity
Variable-Length Results → JSON + List Comprehension

MATCH S=SHORTEST((U:USER)-[*1..1000]-(P:POST))
RETURN U.NAME, P.TITLE,
    [V IN NODES(S) | { CREATOR: V.CREATOR, REG_DATE: V.REG_DATE }] AS S

JSON for variable-length results

- Web-friendly
- Selective
- User "schema"
- Flexible
- Different from SQL types

<table>
<thead>
<tr>
<th>NAME</th>
<th>TITLE</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>{&quot;CREATOR&quot; : &quot;BlueBird21&quot;, &quot;REG_DATE&quot; : 1546300855}, ...]</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
**Variable-Length Results → Table Unnesting**

Current SQL-PG proposal:

```
MATCH SHORTEST((U:USER)-[*1..1000]-(P:POST))
ONE MATCH PER STEP (V, E)
COLUMNS (MATCH_ID, PATH_ID, PATH_SEQ, U.NAME, P.TITLE, V.CREATOR, V.REG_DATE)
```

<table>
<thead>
<tr>
<th>MATCH_ID</th>
<th>PATH_ID</th>
<th>PATH_SEQ</th>
<th>NAME</th>
<th>TITLE</th>
<th>CREATOR</th>
<th>REG_DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>John1973</td>
<td>Graph query languages</td>
<td>?</td>
<td>2019-01-11 15:43:21</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>John1973</td>
<td>Graph query languages</td>
<td>BlueBird21</td>
<td>1546300855</td>
</tr>
</tbody>
</table>

Some type reflection is needed to avoid type errors at runtime.
Variable-Length Results: Aggregation

```
SELECT NAME, TITLE, CREATOR, MIN(REG_DATE)
FROM GRAPH_TABLE(
    MATCH SHORTEST((U:USER)-[*1..1000]-(P:POST))
    ONE MATCH PER STEP (V, E)
    COLUMNS (U.NAME, P.TITLE, V.CREATOR, MATCH_ID, V.REG_DATE)
)
GROUP BY NAME, TITLE, CREATOR, MATCH_ID;
```

```
SELECT NAME, TITLE, CREATOR, REG_DATE
FROM GRAPH_TABLE(
    MATCH S AS SHORTEST((U:USER)-[*1..1000]-(P:POST))
    COLUMNS (U.NAME, P.TITLE, V.CREATOR,
             MIN([V IN VERTICES(S) | V.REG_DATE]) AS REG_DATE)
);
Declarative Beyond Pattern Matching

✓ Pattern matching
✓ Shortest path
✓ All shortest paths
✓ Top-K shortest paths
✗ Community detection / partitioning
✗ Value propagation
✗ Centrality
✗ …
Thank you.

Contact information:

Romans Kasperovics
Developer, SAP HANA Graph
SAP SE, Walldorf (Germany)

[firstname].[lastname]@sap.com