Graph Pattern Matching in SAP HANA

Marcus Paradies, SAP
February 8, 2017
SAP HANA
A big data platform

<table>
<thead>
<tr>
<th>APPLICATION SERVICES</th>
<th>PROCESSING SERVICES</th>
<th>INTEGRATION &amp; QUALITY SERVICES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Server</td>
<td>Spatial</td>
<td>Data Virtualization</td>
</tr>
<tr>
<td>JavaScript</td>
<td>Graph</td>
<td>ELT &amp; Replication</td>
</tr>
<tr>
<td>Fiori UX</td>
<td>Predictive</td>
<td></td>
</tr>
<tr>
<td>Graphic Modeler</td>
<td>Search</td>
<td></td>
</tr>
<tr>
<td>Application Lifecycle Management</td>
<td>Text Analytics</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Streaming Analytics</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Series Data</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Business Functions</td>
<td></td>
</tr>
<tr>
<td>DATABASE SERVICES</td>
<td>Data Quality</td>
<td></td>
</tr>
<tr>
<td>Columnar</td>
<td>Hadoop &amp; Spark Integration</td>
<td></td>
</tr>
<tr>
<td>OLTP+OLAP</td>
<td>Remote Data Sync</td>
<td></td>
</tr>
<tr>
<td>Multi-Core &amp; Parallelization</td>
<td>Openness</td>
<td></td>
</tr>
<tr>
<td>Advanced Compression</td>
<td>Admin &amp; Security</td>
<td></td>
</tr>
<tr>
<td>Multi-tenancy</td>
<td>High Availability &amp; Disaster</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Recovery</td>
<td></td>
</tr>
</tbody>
</table>

- Offers advanced features for graph, text, geospatial, and machine learning
Graph Representation in SAP HANA

Example Graph

- **User**
  - ID: 1
  - Name: 'John'
  - Type: User
  - Rating: 5.0

- **Product**
  - ID: 2
  - Name: 'Shining'
  - Type: Product

- **Category**
  - ID: 3
  - Name: 'Horror'
  - Type: Category

- **Product**
  - ID: 4
  - Name: 'It'
  - Type: Product

**Vertex Table**

<table>
<thead>
<tr>
<th>ID</th>
<th>TYPE</th>
<th>NAME</th>
<th>TITLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>User</td>
<td>John</td>
<td>?</td>
</tr>
<tr>
<td>2</td>
<td>Product</td>
<td>?</td>
<td>Shining</td>
</tr>
<tr>
<td>3</td>
<td>Product</td>
<td>?</td>
<td>It</td>
</tr>
<tr>
<td>4</td>
<td>Category</td>
<td>Horror</td>
<td>?</td>
</tr>
</tbody>
</table>

**Edge Table**

<table>
<thead>
<tr>
<th>ID</th>
<th>SOURCE</th>
<th>TARGET</th>
<th>TYPE</th>
<th>RATING</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
<td>belongsTo</td>
<td>?</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td>belongsTo</td>
<td>?</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>rated</td>
<td>4.0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>3</td>
<td>rated</td>
<td>5.0</td>
</tr>
</tbody>
</table>
How to Consume Graph?

1. Vertex and Edge Tables

```sql
CREATE COLUMN TABLE "MYSCHEMA"."NODES" (  
    ID VARCHAR(100) PRIMARY KEY,  
    TYPE VARCHAR(100),  
    NAME VARCHAR(100),  
    TITLE VARCHAR(100)  
);  

CREATE COLUMN TABLE "MYSCHEMA"."EDGES" (  
    ID INTEGER PRIMARY KEY,  
    SOURCE VARCHAR(100) NOT NULL  
        REFERENCES "MYSCHEMA"."NODES" (ID),  
    TARGET VARCHAR(100) NOT NULL  
        REFERENCES "MYSCHEMA"."NODES" (ID),  
    TYPE VARCHAR(50),  
    RATING FLOAT  
);  
```

2. Create Graph Workspace

```sql
CREATE GRAPH WORKSPACE "MYSCHEMA"."MYGRAPH"  
    EDGE TABLE "MYSCHEMA"."RELATIONSHIPS"  
        SOURCE COLUMN SOURCE  
        TARGET COLUMN TARGET  
        KEY COLUMN ID  
    VERTEX TABLE "MYSCHEMA"."NODES"  
        KEY COLUMN ID;
```

"MYSCHEMA"."MYGRAPH"  
  EDGES  
  NODES
Graph Querying Paradigms

Graph Pattern Matching

"Retrieve all suppliers of Company D"

Graph Analysis

"Compute all communities in the graph"
SAP HANA Graph Architecture

- Interface
  - openCypher
  - GraphScript
- Optimizer
- Plan Generation
- Execution Engine
- Storage
  - Relational Tables

Can use additional secondary graph index structures
openCypher in SAP HANA Graph

**Basic Graph Pattern Matching**
- Vertex topology constraints
- Edge topology constraints

**Attribute Constraints**
- Relational operators: =, <=, >=, <, <>
- Logical operators: AND, OR, NOT

**Solution Modifiers**
- Projection
- Ordering
- Limit
- Skip

**Path Expressions**
- Directed/undirected
- Edge constraints
openCypher Examples

MATCH (a)-[e1]->(b), (a)-[e2]->(b)
WHERE e1.type = 'creates' AND e2.type = 'likes'
RETURN a.name AS name, b.content AS content
ORDER BY a.name ASC
LIMIT 10

MATCH p = (b)-[*1..3]->(a), (b)-[e]->(c)
WHERE a.name = 'Franziska Schwarz'
AND ALL(edges IN RELATIONSHIPS(p) WHERE edges.type = 'follows')
AND e.type = 'creates'
RETURN c.content AS content

Graph-specific operators

Relational operators

For details see SAP HANA Graph Reference
### Some Observations

<table>
<thead>
<tr>
<th>Data Types</th>
<th>Query Composition</th>
<th>Language Semantics</th>
<th>Multiple Graphs</th>
</tr>
</thead>
</table>
| • Customers demand more data types  
  • e.g., timestamp, date, numerical types with higher precision | • Important for views and subqueries | • Matching/Path finding semantics have large impact on performance  
  • Orthogonal feature sets | • Graph referencing by name/identifier  
  • Handling of multiple graph instances in single query |
Thank you

Find out more:

SAP HANA Graph Reference
SAP HANA Academy

SAP HANA, express edition
SAP HANA, express edition FAQ