

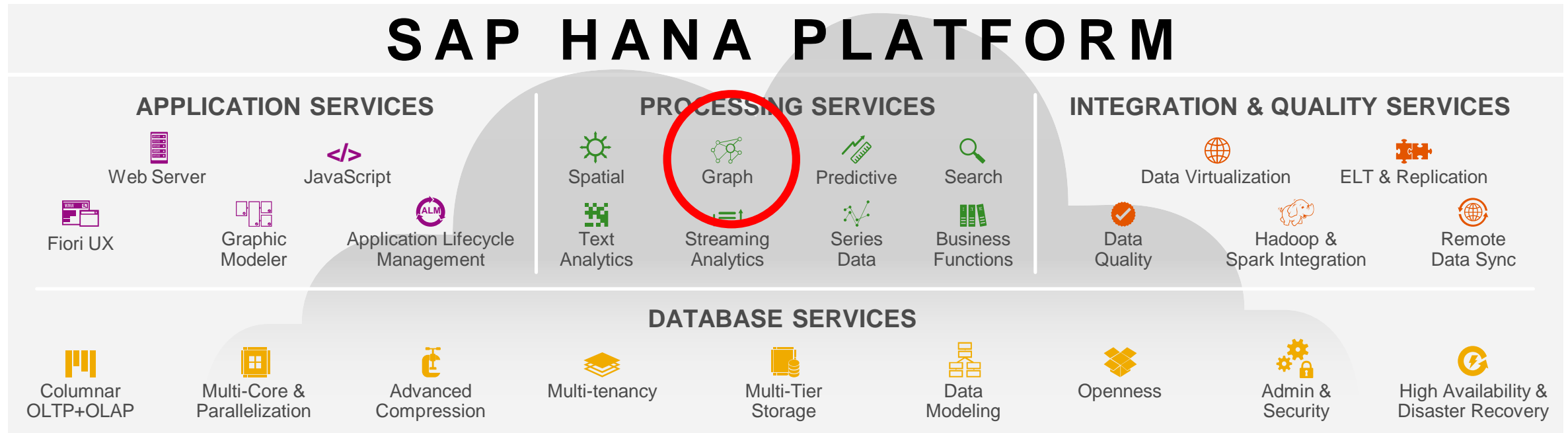


GraphScript: Implementing Complex Graph Algorithms in SAP HANA

Marcus Paradies, Cornelia Kinder, Jan Bross, Thomas Fischer, Romans Kasperovics and Hinnerk Gildhoff, SAP SE
September 1, 2017

PUBLIC

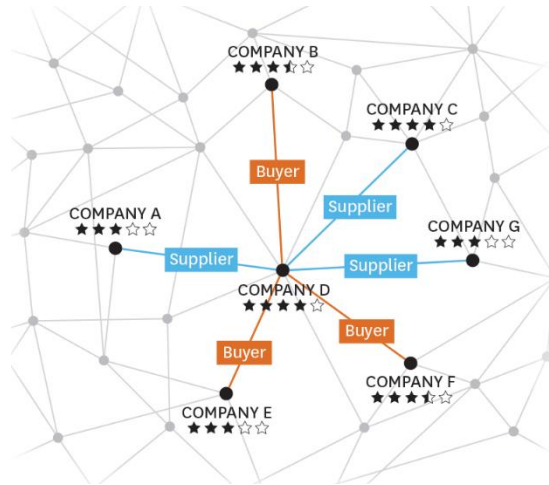
SAP HANA Overview



- Offers advanced analytics features for graph, text, geospatial, and machine learning directly on business data

Graph Querying Paradigms in SAP HANA

Graph Pattern Matching



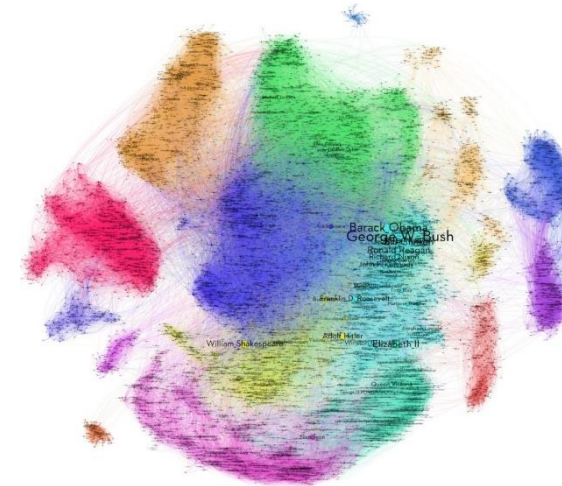
Example Query

"Retrieve all suppliers of Company D"

Language Interface

openCypher*

Graph Analysis



"Compute all communities in the graph"

GraphScript

Design Principles

Expressiveness & Simplicity

- Easy-to-use for graph algorithm implementers
- Support for a large variety of graph algorithm classes and workflows

Minimality & Orthogonality

- Limited but effective set of types and operations thereon
- Extensibility of built-in graph operators

Native Graph Abstraction

- Native exposure of graph-specific types
- Full exposure of graph data model
- Relational only for returning complex results

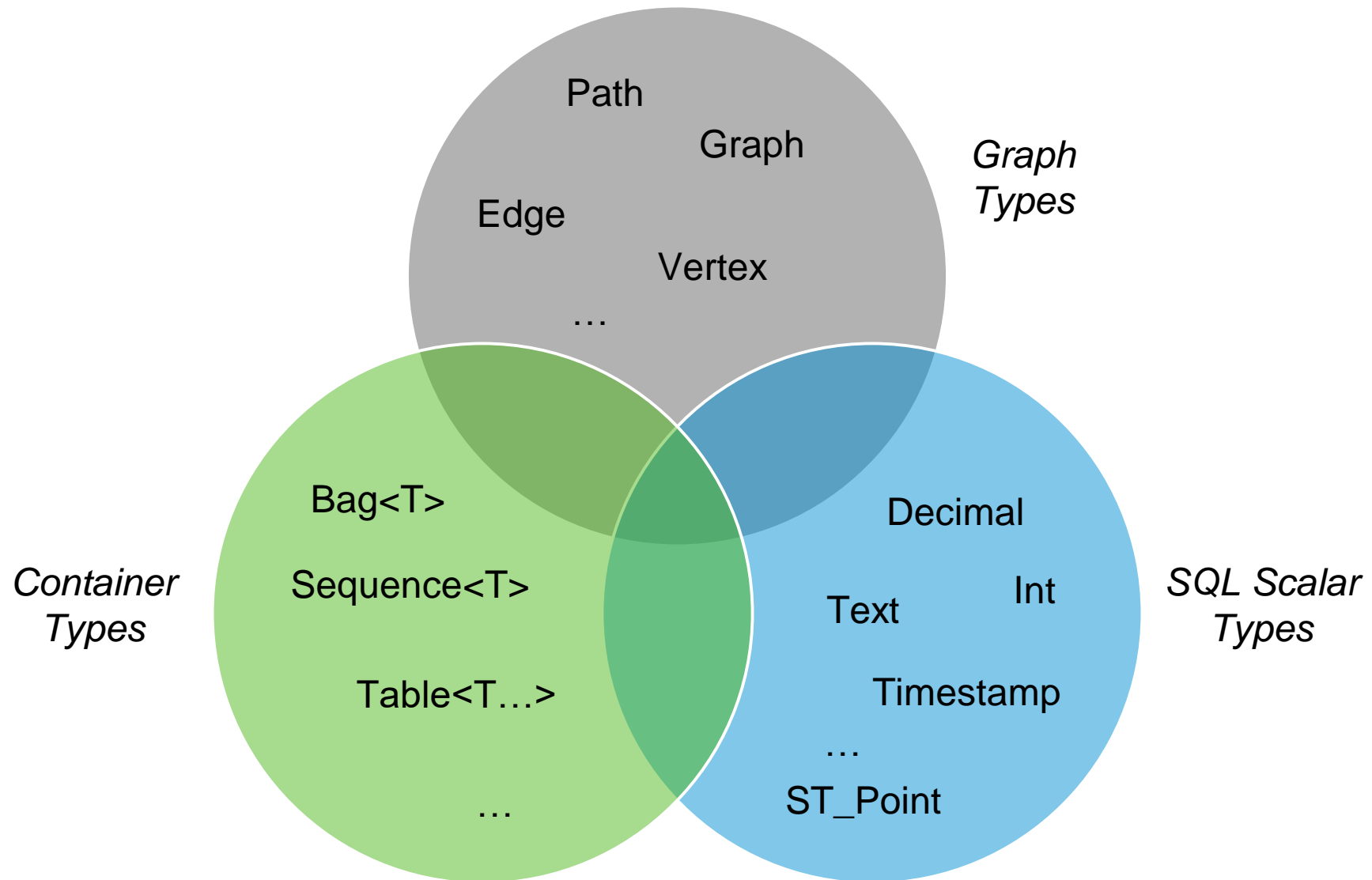
Tight Integration

- Pushdown of operations to relational store
- Reuse of dependency management
- Reuse of resource management

High Performance

- Desired performance close to hand-written code
- Explicit parallelization
- Effective Program Rewritings

GraphScript Type System



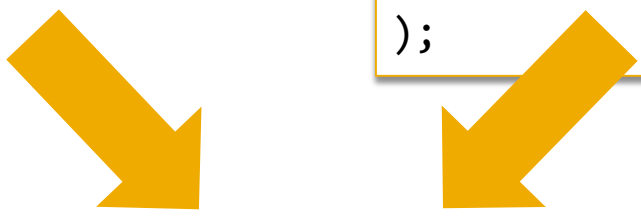
Graph Data Exposure in GraphScript

Vertex Table

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

Edge Table

```
CREATE COLUMN TABLE MYSCHEMA.EDGES (  
  ID INTEGER PRIMARY KEY,  
  SRC VARCHAR(100) NOT NULL  
    REFERENCES MYSCHEMA.VERTICES (ID)  
  TRGT VARCHAR(100) NOT NULL  
    REFERENCES MYSCHEMA.VERTICES (ID)  
  TYPE VARCHAR(50)  
);
```



```
CREATE GRAPH WORKSPACE MYSCHEMA.MY_GRAPH  
  EDGE TABLE MYSCHEMA.EDGES  
    SOURCE COLUMN SRC  
    TARGET COLUMN TRGT  
    KEY COLUMN ID  
  VERTEX TABLE MYSCHEMA.VERTICES  
    KEY COLUMN ID;
```

**Graph Workspace
Metadata Object**

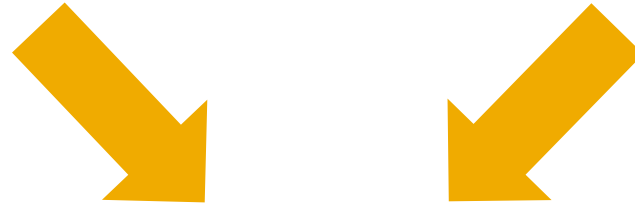
Graph Data Exposure in GraphScript /2

Vertex Table View

```
CREATE VIEW MYSCHEMA.VERTEX_VIEW AS
  SELECT * FROM MYSCHEMA.VERTICES
  WHERE TYPE = 'Person';
```

Edge Table View

```
CREATE VIEW MYSCHEMA.EDGE_VIEW AS
  SELECT * FROM MYSCHEMA.EDGES
  WHERE TYPE = 'knows';
```



```
CREATE GRAPH WORKSPACE MYSCHEMA.MY_SUBGRAPH
  EDGE TABLE MYSCHEMA.EDGE_VIEW
  SOURCE COLUMN SRC
  TARGET COLUMN TRGT
  KEY COLUMN ID
  VERTEX TABLE MYSCHEMA.VERTEX_VIEW
  KEY COLUMN ID;
```

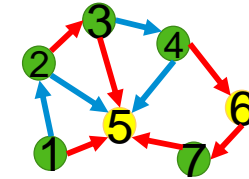
**Graph Workspace
Metadata Object**

A Simple GraphScript Example

```
CREATE PROCEDURE "myGraphProc"(OUT numNeighbors BIGINT)
LANGUAGE GRAPH READS SQL DATA AS
BEGIN
    Graph g = Graph("mySchema","myGraph");
    ALTER g ADD TEMPORARY VERTEX ATTRIBUTE(BIGINT cnt = 0);
    FOREACH v IN Vertices(:g) {
        v.cnt = Count(Neighbors(:g, :v, 1, 3));
    }
    FOREACH v IN Vertices(:g) {
        numNeighbors += :v.cnt;
    }
END
```


Inducing Subgraphs

Data graph:



<p><i>"Induce a graph over all blue edges"</i></p>		<pre>Graph g = Subgraph(:g, e IN Edges(:g) WHERE :e.color == 'blue');</pre>
<p><i>"Induce a graph over all red edges that connect a green and a yellow vertex"</i></p>		<pre>Graph g = Subgraph(:g, e IN Edges(:g) WHERE Source(:e).color == 'green' AND Target(:e).color == 'yellow' AND :e.color == 'red');</pre>
<p><i>"Induce a graph over all all vertices that are reachable from vertex 4"</i></p>		<pre>Vertex v1 = Vertex(:g, 4); Graph g = Subgraph(:g, v IN Vertices(:g) WHERE IS_REACHABLE(:g, :v1, :v);</pre>

Integration with other Data Models/Scalar Types

Creation of Relational Output from GraphScript

```
Graph g = Graph("myWorkspace");
ALTER g ADD TEMPORARY VERTEX ATTRIBUTE(DOUBLE length = 0);
FOREACH v IN Vertices(:g) {
    Path p = Shortest_Path(:g, :v, Vertex(:g, 1));
    v.length = Length(:p);
}
outTab = SELECT :v.id, :v.length FOREACH v IN Vertices(:g);
```

Integration with Geospatial Processing

```
Graph g = Graph("myWorkspace");
ST_Geometry area = Vertex(:g, 'Munich').area;
Graph g1 = Subgraph(:g, v IN Vertices(:g) WHERE :v.type == 'Person'
                    AND ST_Within(:v.location , :area));
```

Conclusion

Language Constructs

- Rich type system with native graph types
- Powerful imperative constructs

Code Generation

- Generation of low-level code against internal Graph Storage interface
- Elimination of query processing on external vertex/edge keys
- Pushdown of filter conditions to relational engine

Future Work

- More language extensions towards fast traversals and user-defined function invocations
- More advanced GraphScript program rewritings and optimizations

Thank you.

Contact information:

Marcus Paradies

marcus.paradies@sap.com

© 2017 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company.

The information contained herein may be changed without prior notice. Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE's or its affiliated companies' strategy and possible future developments, products, and/or platform directions and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, and they should not be relied upon in making purchasing decisions.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies.

See <http://global.sap.com/corporate-en/legal/copyright/index.epx> for additional trademark information and notices.