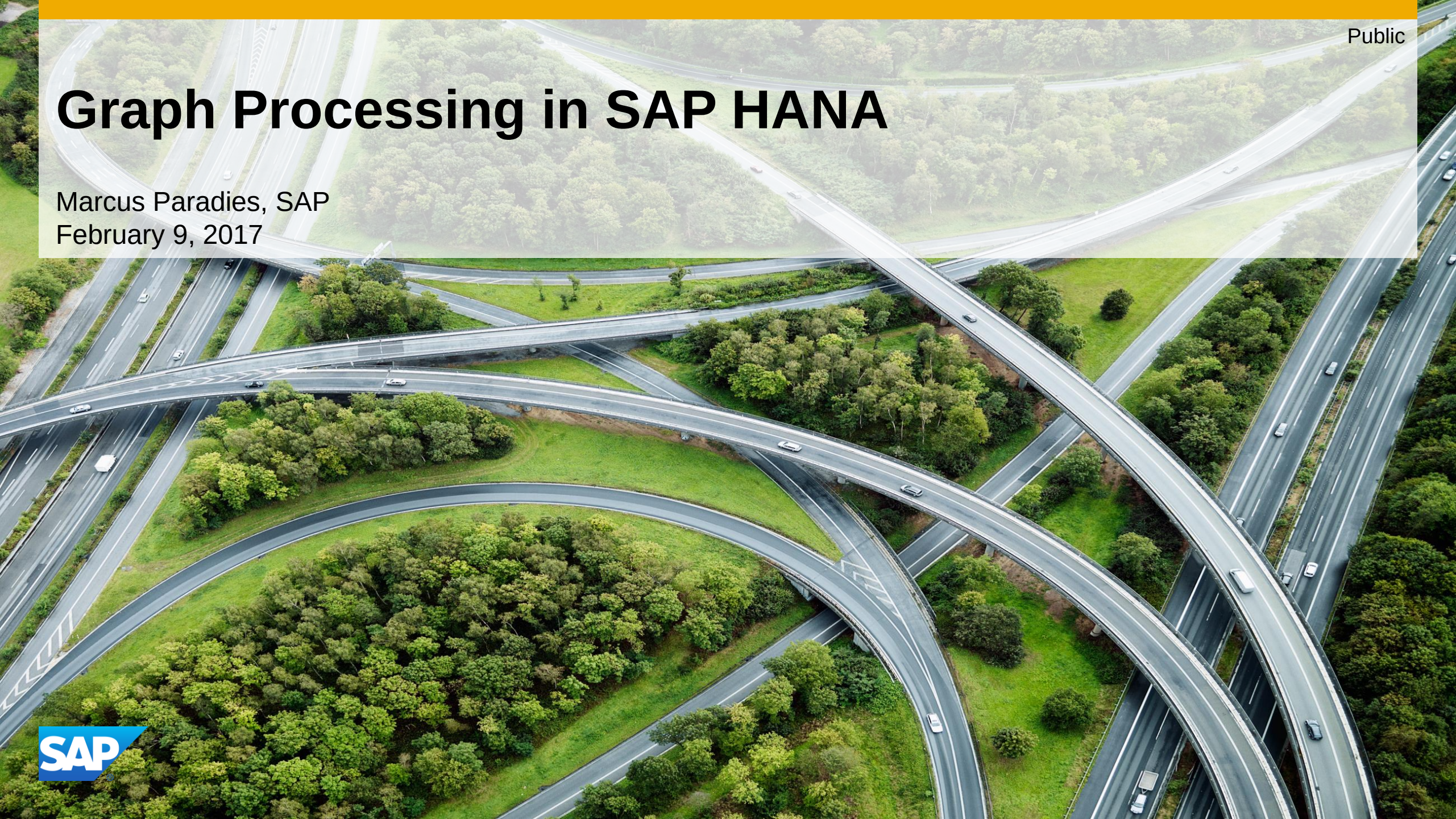
Public

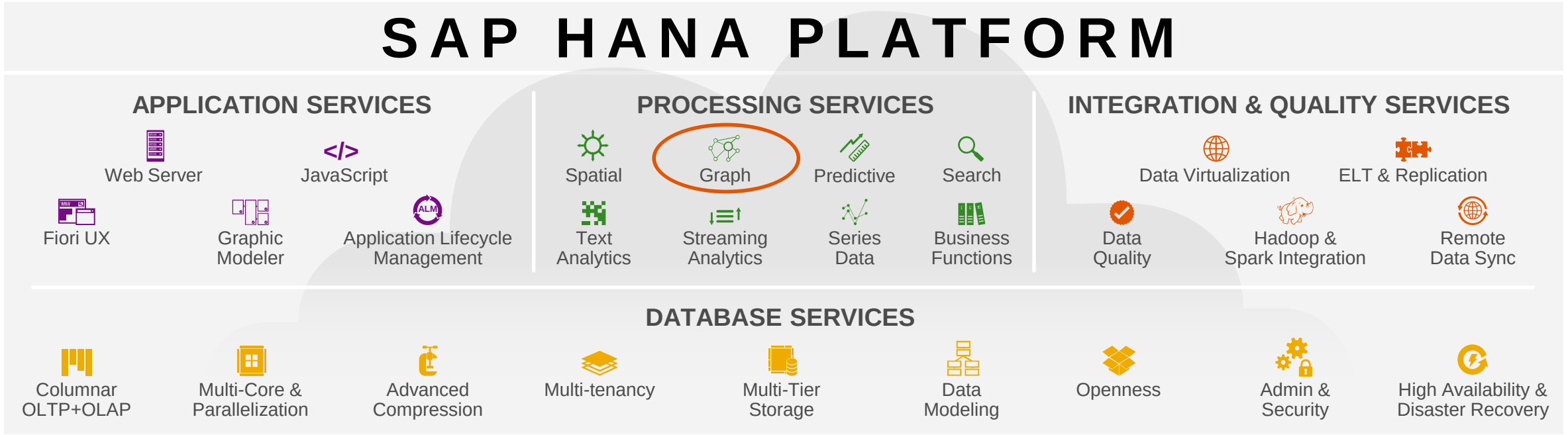
# Graph Processing in SAP HANA

Marcus Paradies, SAP
February 9, 2017



# SAP HANA

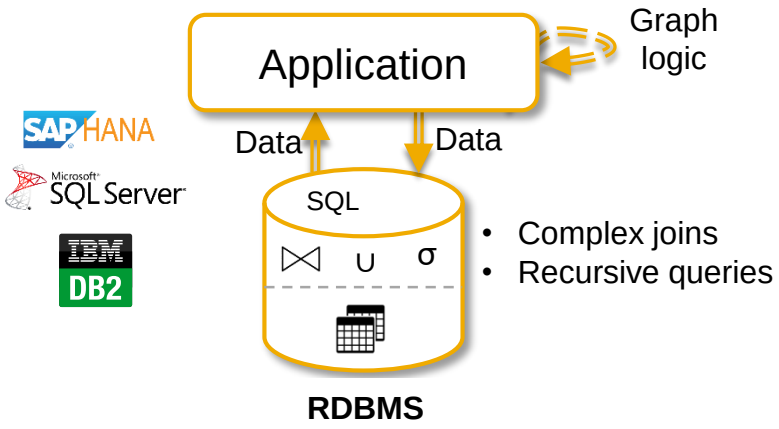
## A big data platform




- Offers advanced features for graph, text, geospatial, and machine learning

# Graph Processing on Enterprise Data

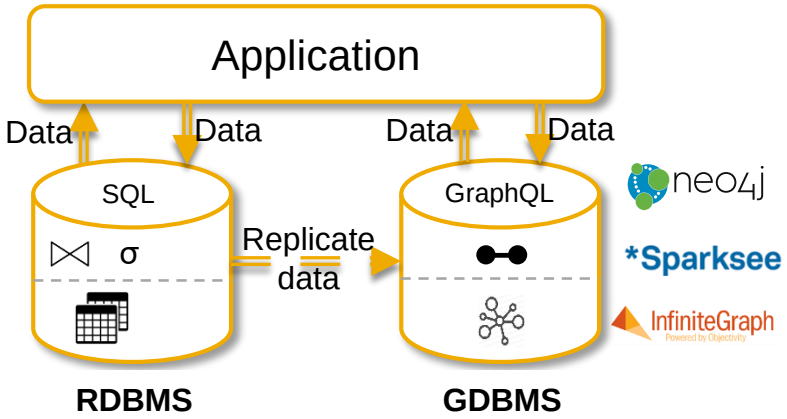
## Relational + Application Logic




✚ Consistent view of the data

➖ SQL not suitable for graph processing
➖ No graph abstraction
➖ Data transfer to application

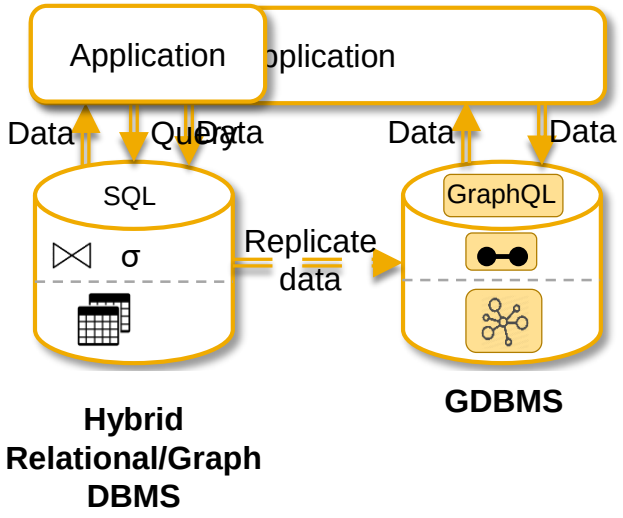
## Relational + Graph + Application Logic



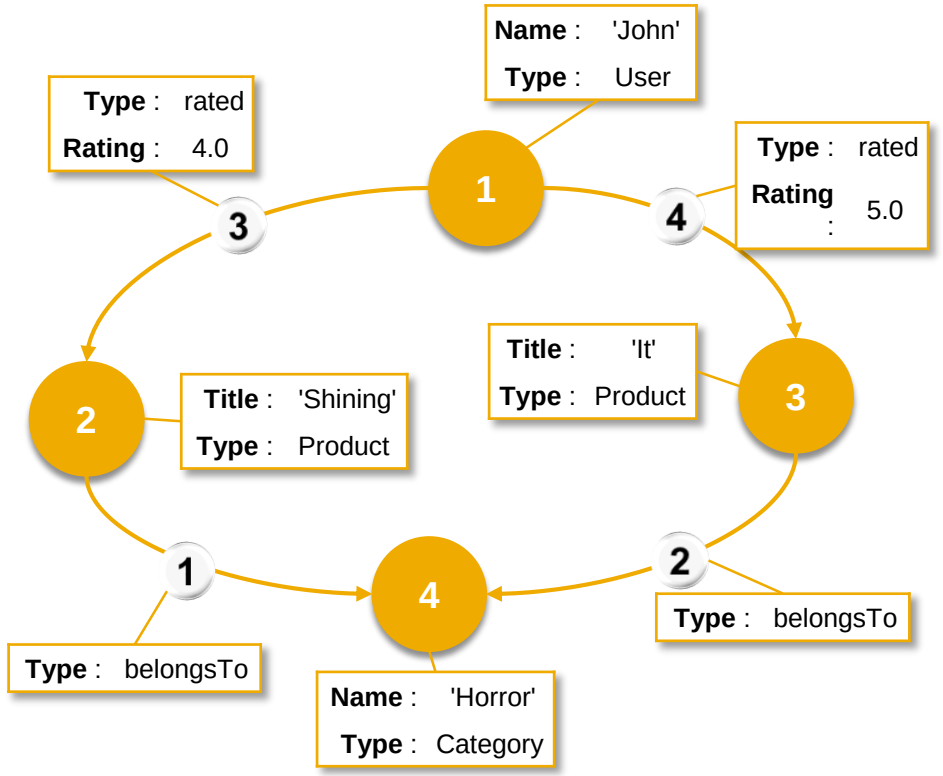

✚ Query performance
✚ Graph abstraction

➖ No online processing
➖ Data transfer to application
➖ Federated query processing

# Graph Processing on Enterprise Data



# Graph Representation in SAP HANA




**Example Graph**

| ID | TYPE | NAME | TITLE |
|---|---|---|---|
| 1 | User | John | ? |
| 2 | Product | ? | Shining |
| 3 | Product | ? | It |
| 4 | Category | Horror | ? |

**Vertex column group**

**Vertex Table**

| ID | SOURCE | TARGET | TYPE | RATING |
|---|---|---|---|---|
| 1 | 2 | 4 | belongsTo | ? |
| 2 | 3 | 4 | belongsTo | ? |
| 3 | 1 | 2 | belongsTo | ? |
| 4 | 1 | 3 | rated | 4.0 |
| 5 | 2 | 4 | rated | 5.0 |
| 6 | 1 | 2 | rated | 4.0 |
| 7 | 1 | 3 | rated | 5.0 |

**Edge column group**

**Edge Table**

# How to Consume Graph?

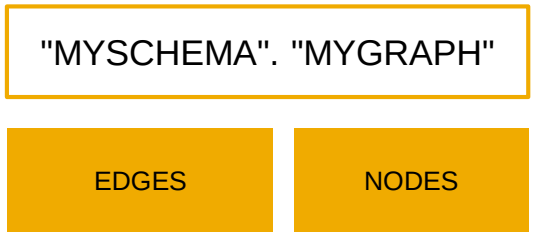
## 1. Vertex and Edge Tables

```
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
);
```

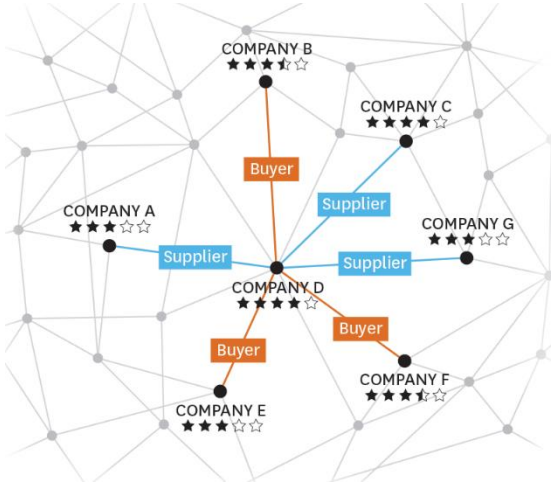
## 2. Create Graph Workspace

```
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;
```



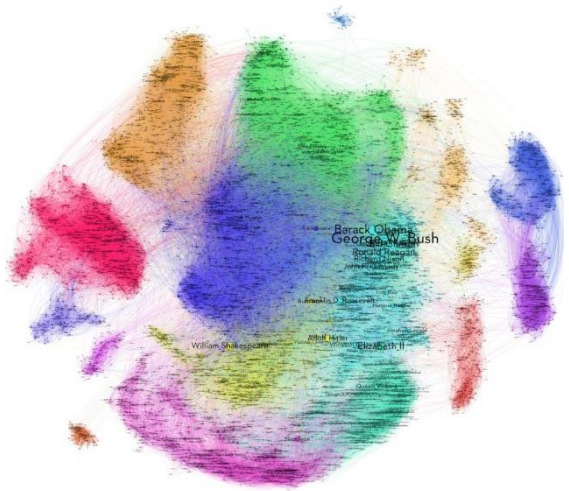
# Graph Querying Paradigms

## Graph Pattern Matching



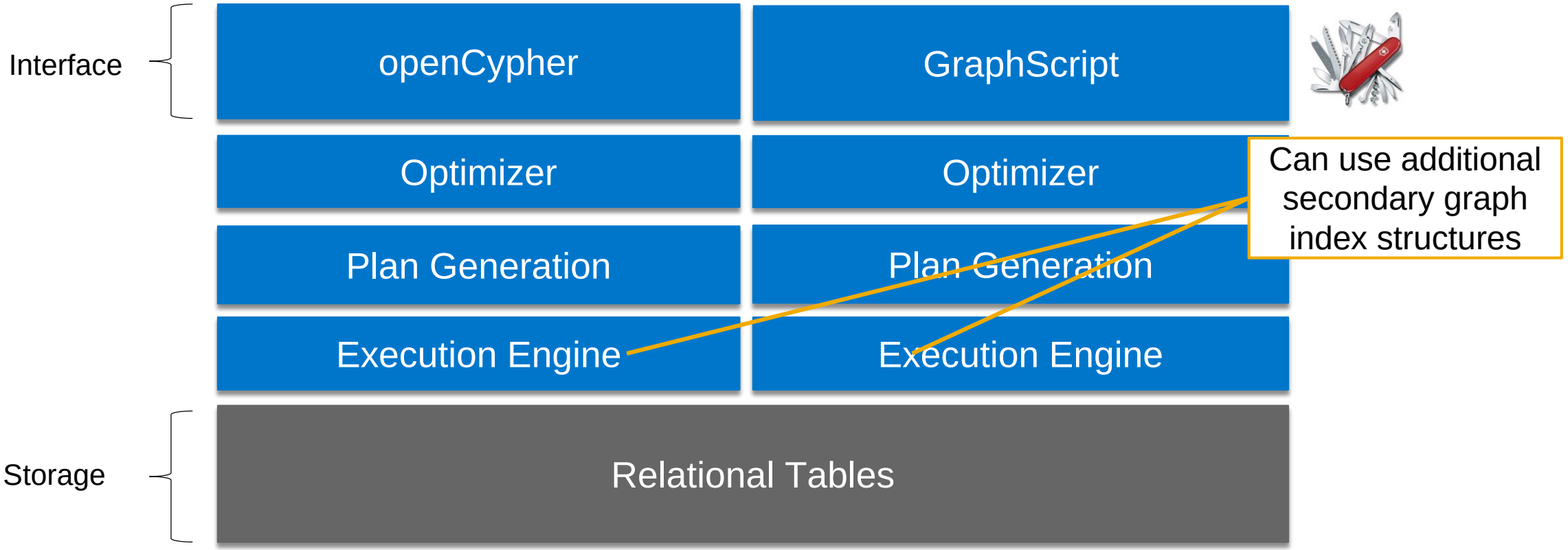

"Retrieve all suppliers of Company D"

## Graph Analysis



"Compute all communities in the graph"

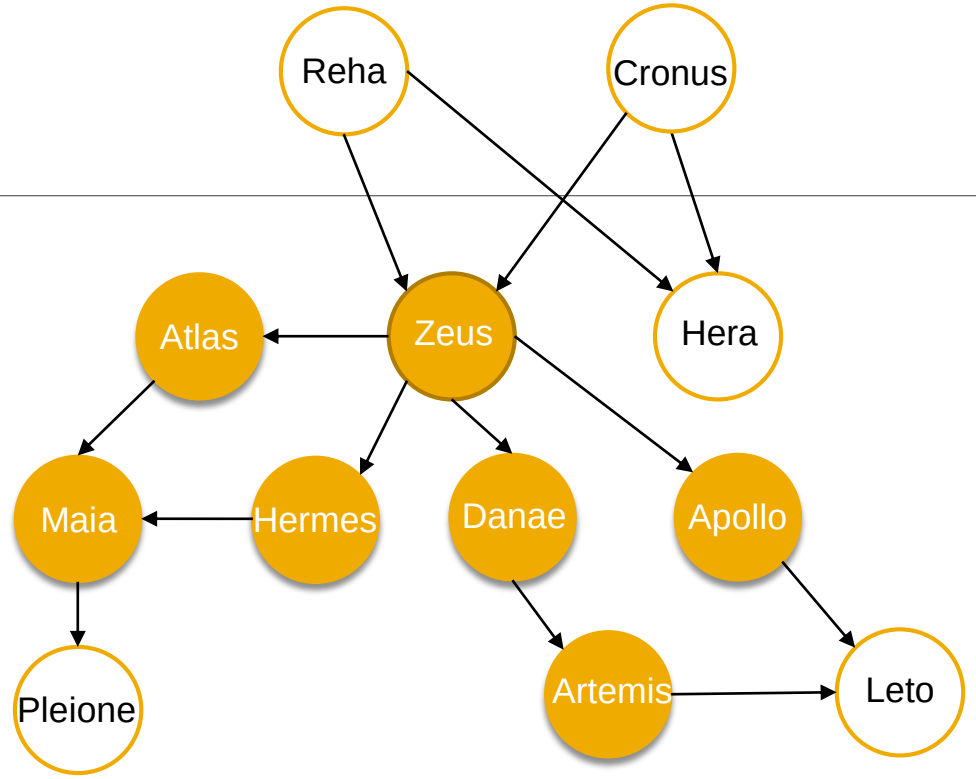
# SAP HANA Graph Architecture



# Neigborhood Search

## Graph Traversal

- Search for neighborhood
- Input Parameters
  - Start vertices
  - Search direction (*)
  - Min/max depth (*)
  - Vertex/edge filter (*)
- Output
  - Vertex Key
  - Search Depth




**Example:**

```
SELECT * FROM GREEK.NEIGHBORHOOD
WITH PARAMETERS (
    'placeholder' = ('$startVertices$', [‘zeus‘]),
    'placeholder' = ('$minDepth$',‘0')
    'placeholder' = ('$maxDepth$',‘2')
    'placeholder' = ('$edgeFilter$',‘rel=parentOf'));
```

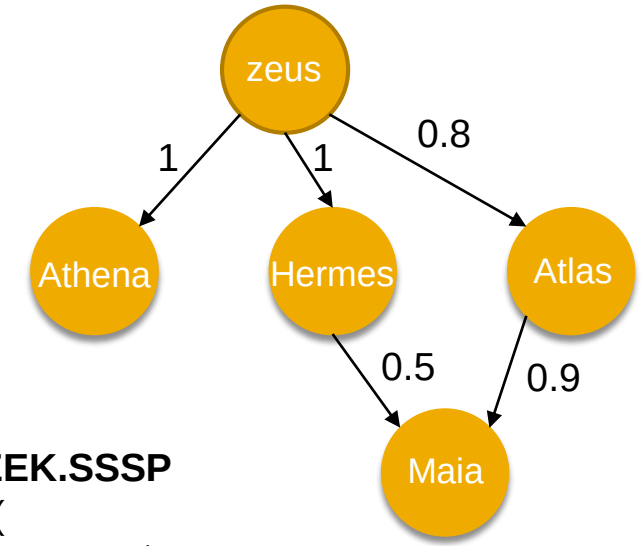
| Name | Depth |
|---|---|
| Atlas | 1 |
| Hermes | 1 |
| Danae | 1 |
| Apollo | 1 |
| Maia | 2 |
| Artemis | 2 |

* := optional parameter

# Shortest Path

## Single Source Shortest Path (SSSP)

- Single-Source Shortest Path
  - Provides shortest path to from start vertex to all reachable vertices in the graph
- Input Parameters
  - Start vertex
  - Input weight column (*)
- Output
  - Vertex key
  - Calculated weight
  - Shortest path start to end point




**Example**:

```
SELECT * FROM GREEK.SSSP
WITH PARAMETERS (
    'placeholder' = ('$startVertex$', ‘zeus‘),
    'placeholder' = ('$edgeWeightInputColumn$',‘weight'));
```

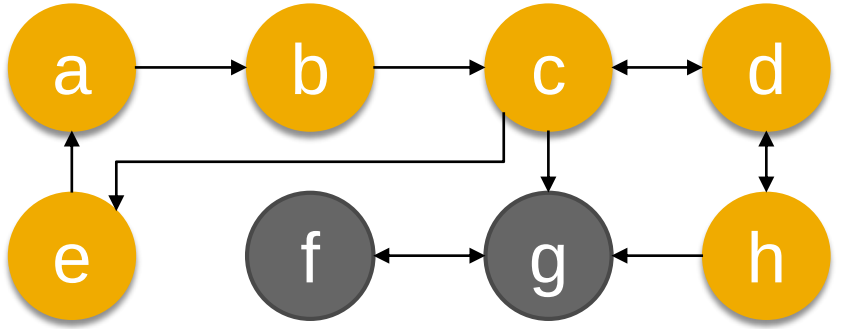
| Name | Weight | Path |
|---|---|---|
| Athena | 1 | Zeus -> Athena |
| Hermes | 1 | Zeus -> Hermes |
| Atlas | 0.8 | Zeus -> Atlas |
| Maia | 1.5 | Zeus -> Maia |

* := optional parameter

# Strongly Connected Components

## SCC

- Search for strongly connected components
- Output
  - Vertex key
  - Component ID




| Vertex | Component |
|---|---|
| a | 1 |
| b | 1 |
| c | 1 |
| d | 1 |
| e | 1 |
| h | 1 |
| f | 2 |
| g | 2 |

**Example:**

SELECT * FROM **MY.SCC;**

* := optional parameter

# GraphScript

## as Stored Procedure Language

- GraphScript is a native stored procedure language of SAP HANA
- Compiled into highly efficient, parallelized code (no translation to SQL)

```
CREATE PROCEDURE graphProc(OUT distance DOUBLE)
LANGUAGE GRAPH READS SQL DATA AS
BEGIN
  GRAPH g = GRAPH("GRAPH_PROC_TEST","MYGRAPH");
  VERTEX v1 = VERTEX(:g, 1);
  VERTEX v2 = VERTEX(:g, 2);
  PATH p = SHORTEST_PATH(:g, :v1, :v2);
  distance = LENGTH(:p);
END
```

# GraphScript

## Type System

### Native exposure of graph-specific data types

- Vertices, edges, paths, (sub)-graphs, and collections

### Support attributes on vertices and edges

- Simple data types: int, double, string
- Complex data types: ST_Point, text

### Efficient transformations between instances of specific types

```
GRAPH g = GRAPH("schema", "name");

VERTEX v = VERTEX(:g, 1);
EDGE e1 = EDGE(:g, 1);

INT inc = :v.income;
ST_POINT p = :e1.location;

MULTISET<VERTEX> vertices = VERTICES(:g);
MULTISET<EDGE> edges = EDGES(:g);

PATH p = SHORTEST_PATH(:g,:v1,:v2);

MULTISET<VERTEX> vertices1 = VERTICES(:p);
MULTISET<EDGE> edges1 = EDGES(:p);
```

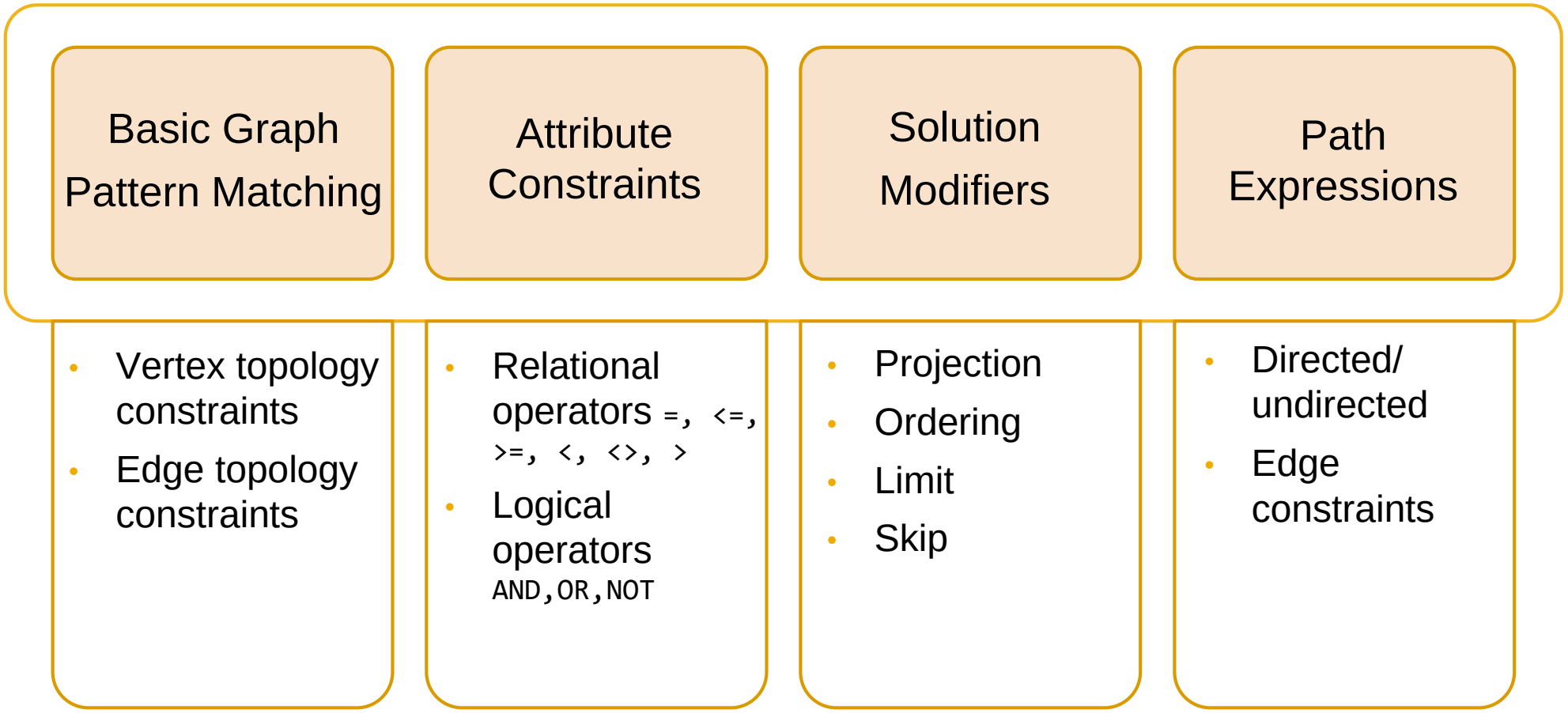
# GraphScript

## Invocation of Built-In Graph Algorithms

- Invocation (and parametrization) of built-in graph algorithms
- All built-in graph algorithms can be called from GraphScript and the result can be post-processed

```
GRAPH g = GRAPH("schema", "name");
GRAPH sg = GRAPH(v IN Vertices(:g) WHERE :v.type == 'Person')
VERTEX v1 = VERTEX(:sg, 1);
VERTEX v2 = VERTEX(:sg, 42);
PATH p = SHORTEST_PATH(:sg, :v1, :v2);
INT sum = 0;
FOREACH e : EDGES(:p) {
   sum += :e.weight;
}
```

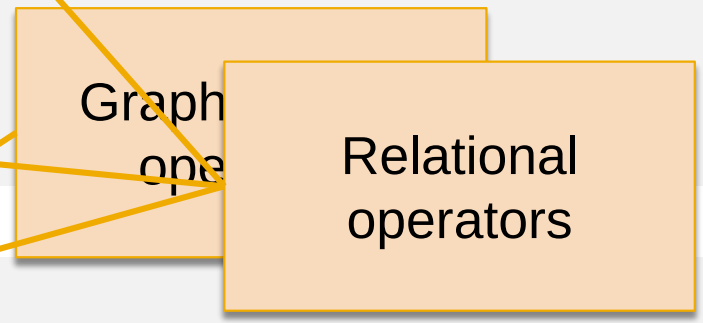
# openCypher in SAP HANA Graph



# 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
```




For details see SAP HANA Graph Reference

# Summary

- **Native graph processing in SAP HANA**
- **Tight integration, reuse where possible, specialize where sensible**
- **Two language interfaces**
  - Graph pattern matching - openCypher
  - Graph Analysis - GraphScript

# Thank you

**Find out more:**

SAP HANA Graph Reference
SAP HANA Academy

SAP HANA, express edition
SAP HANA, express edition FAQ