

## **Graph Processing Using SAP HANA: A Teaching Case**

### **Mark Hwang**

Central Michigan University, USA

Email: [Mark.hwang@cmich.edu](mailto:Mark.hwang@cmich.edu)

### **Abstract**

*The purpose of this teaching case is to develop a hands-on exercise on graph processing using a hybrid system. The paper provides a background on graph databases and how graph processing is supported in a hybrid system, SAP HANA. It also details step-by-step instructions on how to create, modify, and process a graph database using SAP HANA. The teaching case provides instructors with materials that can be readily implemented in classrooms. Because there is no software to purchase or install, there are no costs to the instructor or the school. The assignment can be completed by any student with knowledge of SQL. All the software tools are accessible to any instructor whose school is a member of the SAP University Alliances program, which is free to join.*

**Key words:** *Graph processing; graph databases; NoSQL, SAP HANA; technology.*

**JEL Classification:** I21

**PsycINFO Classification:** 3530

**FoR Code:** 1302; 1503

**ERA Journal ID#:** 35696

## Introduction

Relational databases have been at the core of enterprise systems for the last forty years, supporting applications ranging from sales order processing to supply chain management. In a relational database, structured data are organized neatly into tables made of columns and rows. In the age of big data, however, organizations are inundated with increasingly larger amount and varied data including audios and videos, emails, tweets, and sensor data. The NoSQL market was developed in response to the need to handle unstructured data in various ways; chief among them, key-value pairs, document stores, wide-column stores and graph databases (Yegulalp, 2017). Among the non-relational contenders, graph databases are especially suited for applications where connections among data points are the focus such as finding friends in a social network. In a graph database, data are organized into a network of interconnected nodes. Unlike the relational model where a rigid schema is defined, a graph can be created and modified without a fixed schema (Robinson et al., 2015; Hurlburt et al., 2017). Another advantage of a graph database is traversing the graph or network is extremely fast because all the connections are pre-defined, making it an ideal data store for other applications besides social networking including mapping, route planning and network diagnosis (Mortleman, 2016). Another common application of graph databases is pattern matching that can be readily explored in the connections. The result can be a recommendation to purchase a product or service or an indication of a cyberattack (Elhadi et al., 2014) or fraud as exemplified in the Panama Papers case (McKenna, 2016).

Many fledgling database vendors currently compete in the emerging graph database market. According to one recent estimate, over 50 offerings are available (Longbottom, 2016). Major suppliers of relational database systems, including Oracle, IBM and Microsoft, have also extended their products by either developing a separate graph server or adding graph processing capabilities to a relational data store. The latter approach is appealing as it allows an organization to leverage its existing databases to new use cases that involve graph processing. A hybrid system that combines graph processing with a relational or non-graph database is beneficial as the system scales up and the need for integration of data from different systems arises (Hurlburt et al., 2017). An example hybrid system is SAP HANA. SAP is a leading enterprise software vendor best known for its Enterprise Resource Planning (ERP) products. In recent years SAP developed its proprietary in-memory database known as SAP HANA, which underpins a technology platform that supports real-time enterprise transactional and analytical processing. The underlying data model of SAP HANA is relational, but it also includes a graph engine that supports graph processing (Rudolf et al., 2013). This assignment allows students to practice graph processing using a relational database as detailed in the following sections.

## Learning Outcomes/Assessment

The objective of the exercise is to provide students with a hands-on experience in graph processing. Students are expected to:

- a) Create a graph database using SQL.
- b) Modify the graph database using SQL.
- c) Process the graph database using built-in algorithms of SAP HANA.
- d) Apply the skills to other domains.

The first three learning outcomes can be assessed by the screenshots taken by students after completing each of the required steps as illustrated throughout the assignment.

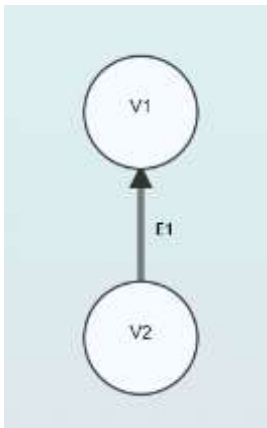
The fourth learning outcome can be assessed by several open-ended questions asking students to reflect and think critically, as noted at the end of the assignment.

## The Assignment

### **SAP HANA Graph Processing<sup>1</sup>**

SAP HANA has a graph engine that supports graph processing of data stored in columnar tables. A graph is basically a network of vertices or nodes interconnected by edges. Vertices represent entities of interest in a specific domain and edges represent their relationships. A simple graph consisting of two vertices and one edge is shown below. In SAP HANA an edge is always “directed”. For example, to record the fact that Mary lives in New York, V2 will represent “Mary” and V1 will represent “New York”. The edge E1 will be “lives in” and will be directed from “Mary” to “New York”. Two vertices can have a reciprocal relationship and thus two directed arrows pointing to each other. For instance, if V2 is “Mary” and V1 is her Instagram pal “Pete,” and they follow each other on the social network, there can be an edge E1 “follows” pointing from V2 to V1 and an edge E2 “follows” pointing from V1 to V2. Both vertices and edges can have properties, facts that are of importance in a specific domain. For example, for people such as Mary and Pete, we may need to know their ages and income levels. Similarly, for places such as New York we may need to track properties such as its population, latitude and longitude, etc. We can also record properties of the edge “lives in” or “follows” such as the starting date of the relationship.

**Figure 1:**  
*A Graph as a Directed Network*



An SAP HANA graph workspace is created to describe a graph, which can then be used to support graph processing of data in response to user query. Common graph processing operations include the shortest path, neighbourhood search, and the strongest connected components (Robinson et al., 2015). Pattern matching is another useful application that allows identification of vertices with similar properties. SAP HANA

<sup>1</sup> This exercise is adopted from [SAP HANA Graph Reference](#).

has built-in algorithms for all common graph processing operations. Users can use all built-in algorithms without writing any code.

This exercise illustrates how easy it is to create a graph in SAP HANA from a vertices table and an edges table. The only prerequisite for completing the assignment is familiarity with SQL, the common language for all relational databases. After the tables are created a graph workspace is created to facilitate graph processing. Just like any tables in a relational database, the graph tables can be modified easily to suit changing user requirements. The students will practice the creation, processing and modification of a graph. Sample data is adopted from the Greek Mythology graph data described in the [SAP HANA Graph Data Model](#), which is replicated on the last page of this document. Each vertex in the graph represents a mythological figure and his or her properties including the name, the type of the figure (e.g., god or titan) and his or her residence. The edges indicate the relationships among the figures (i.e., spousal or parental relationships). For each relationship, the source and target are defined, e.g., Chaos has a daughter named Gaia and, therefore, Chaos is the source and Gaia is the target of an edge named "hasDaughter".

### **Section I: Create the graph**

The following SQL code can be used to create the two tables and the graph workspace:

```
CREATE COLUMN TABLE "MEMBERS" (
    "NAME" VARCHAR(100) PRIMARY KEY,
    "TYPE" VARCHAR(100),
    "RESIDENCE" VARCHAR(100)
);

INSERT INTO "MEMBERS"("NAME", "TYPE") VALUES ('Chaos', 'primordial deity');
INSERT INTO "MEMBERS"("NAME", "TYPE") VALUES ('Gaia', 'primordial deity');
INSERT INTO "MEMBERS"("NAME", "TYPE") VALUES ('Uranus', 'primordial deity');
INSERT INTO "MEMBERS"("NAME", "TYPE", "RESIDENCE") VALUES ('Rhea', 'titan', 'Tartarus');
INSERT INTO "MEMBERS"("NAME", "TYPE", "RESIDENCE") VALUES ('Cronus', 'titan', 'Tartarus');
INSERT INTO "MEMBERS"("NAME", "TYPE", "RESIDENCE") VALUES ('Zeus', 'god', 'Olymp');
INSERT INTO "MEMBERS"("NAME", "TYPE", "RESIDENCE") VALUES ('Poseidon', 'god', 'Olymp');
INSERT INTO "MEMBERS"("NAME", "TYPE", "RESIDENCE") VALUES ('Hades', 'god', 'Underworld');
INSERT INTO "MEMBERS"("NAME", "TYPE", "RESIDENCE") VALUES ('Hera', 'god', 'Olymp');
INSERT INTO "MEMBERS"("NAME", "TYPE", "RESIDENCE") VALUES ('Demeter', 'god', 'Olymp');
INSERT INTO "MEMBERS"("NAME", "TYPE", "RESIDENCE") VALUES ('Athena', 'god', 'Olymp');
INSERT INTO "MEMBERS"("NAME", "TYPE", "RESIDENCE") VALUES ('Ares', 'god', 'Olymp');
INSERT INTO "MEMBERS"("NAME", "TYPE", "RESIDENCE") VALUES ('Aphrodite', 'god', 'Olymp');
INSERT INTO "MEMBERS"("NAME", "TYPE", "RESIDENCE") VALUES ('Hephaestus', 'god', 'Olymp');
INSERT INTO "MEMBERS"("NAME", "TYPE", "RESIDENCE") VALUES ('Persephone', 'god',
'Underworld');

CREATE COLUMN TABLE "RELATIONSHIPS" (
    "KEY" INT UNIQUE NOT NULL,
    "SOURCE" VARCHAR(100) NOT NULL REFERENCES "MEMBERS" ("NAME") ON UPDATE
CASCADE ON DELETE CASCADE,
    "TARGET" VARCHAR(100) NOT NULL REFERENCES "MEMBERS" ("NAME") ON UPDATE
CASCADE ON DELETE CASCADE,
    "TYPE" VARCHAR(100)
```

);

```

INSERT INTO "RELATIONSHIPS"("KEY", "SOURCE", "TARGET", "TYPE") VALUES (1, 'Chaos', 'Gaia',
'hasDaughter');
INSERT INTO "RELATIONSHIPS"("KEY", "SOURCE", "TARGET", "TYPE") VALUES (2, 'Gaia',
'Uranus', 'hasSon');
INSERT INTO "RELATIONSHIPS"("KEY", "SOURCE", "TARGET", "TYPE") VALUES (3, 'Gaia',
'Cronus', 'hasSon');
INSERT INTO "RELATIONSHIPS"("KEY", "SOURCE", "TARGET", "TYPE") VALUES (4, 'Uranus',
'Cronus', 'hasSon');
INSERT INTO "RELATIONSHIPS"("KEY", "SOURCE", "TARGET", "TYPE") VALUES (5, 'Gaia', 'Rhea',
'hasDaughter');
INSERT INTO "RELATIONSHIPS"("KEY", "SOURCE", "TARGET", "TYPE") VALUES (6, 'Uranus',
'Rhea', 'hasDaughter');
INSERT INTO "RELATIONSHIPS"("KEY", "SOURCE", "TARGET", "TYPE") VALUES (7, 'Cronus',
'Zeus', 'hasSon');
INSERT INTO "RELATIONSHIPS"("KEY", "SOURCE", "TARGET", "TYPE") VALUES (8, 'Rhea', 'Zeus',
'hasSon');
INSERT INTO "RELATIONSHIPS"("KEY", "SOURCE", "TARGET", "TYPE") VALUES (9, 'Cronus',
'Hera', 'hasDaughter');
INSERT INTO "RELATIONSHIPS"("KEY", "SOURCE", "TARGET", "TYPE") VALUES (10, 'Rhea',
'Hera', 'hasDaughter');
INSERT INTO "RELATIONSHIPS"("KEY", "SOURCE", "TARGET", "TYPE") VALUES (11, 'Cronus',
'Demeter', 'hasDaughter');
INSERT INTO "RELATIONSHIPS"("KEY", "SOURCE", "TARGET", "TYPE") VALUES (12, 'Rhea',
'Demeter', 'hasDaughter');
INSERT INTO "RELATIONSHIPS"("KEY", "SOURCE", "TARGET", "TYPE") VALUES (13, 'Cronus',
'Poseidon', 'hasSon');
INSERT INTO "RELATIONSHIPS"("KEY", "SOURCE", "TARGET", "TYPE") VALUES (14, 'Rhea',
'Poseidon', 'hasSon');
INSERT INTO "RELATIONSHIPS"("KEY", "SOURCE", "TARGET", "TYPE") VALUES (15, 'Cronus',
'Hades', 'hasSon');
INSERT INTO "RELATIONSHIPS"("KEY", "SOURCE", "TARGET", "TYPE") VALUES (16, 'Rhea',
'Hades', 'hasSon');
INSERT INTO "RELATIONSHIPS"("KEY", "SOURCE", "TARGET", "TYPE") VALUES (17, 'Zeus',
'Athena', 'hasDaughter');
INSERT INTO "RELATIONSHIPS"("KEY", "SOURCE", "TARGET", "TYPE") VALUES (18, 'Zeus', 'Ares',
'hasSon');
INSERT INTO "RELATIONSHIPS"("KEY", "SOURCE", "TARGET", "TYPE") VALUES (19, 'Hera', 'Ares',
'hasSon');
INSERT INTO "RELATIONSHIPS"("KEY", "SOURCE", "TARGET", "TYPE") VALUES (20, 'Uranus',
'Aphrodite', 'hasDaughter');
INSERT INTO "RELATIONSHIPS"("KEY", "SOURCE", "TARGET", "TYPE") VALUES (21, 'Zeus',
'Hephaestus', 'hasSon');
INSERT INTO "RELATIONSHIPS"("KEY", "SOURCE", "TARGET", "TYPE") VALUES (22, 'Hera',
'Hephaestus', 'hasSon');
INSERT INTO "RELATIONSHIPS"("KEY", "SOURCE", "TARGET", "TYPE") VALUES (23, 'Zeus',
'Persephone', 'hasDaughter');
INSERT INTO "RELATIONSHIPS"("KEY", "SOURCE", "TARGET", "TYPE") VALUES (24, 'Demeter',
'Persephone', 'hasDaughter');
INSERT INTO "RELATIONSHIPS"("KEY", "SOURCE", "TARGET", "TYPE") VALUES (25, 'Zeus', 'Hera',
'marriedTo');
INSERT INTO "RELATIONSHIPS"("KEY", "SOURCE", "TARGET", "TYPE") VALUES (26, 'Hera', 'Zeus',
'marriedTo');
INSERT INTO "RELATIONSHIPS"("KEY", "SOURCE", "TARGET", "TYPE") VALUES (27, 'Hades',
'Persephone', 'marriedTo');

```

```

INSERT INTO "RELATIONSHIPS"("KEY", "SOURCE", "TARGET", "TYPE") VALUES (28, 'Persephone',
'Hades', 'marriedTo');
INSERT INTO "RELATIONSHIPS"("KEY", "SOURCE", "TARGET", "TYPE") VALUES (29, 'Aphrodite',
'Hephaestus', 'marriedTo');
INSERT INTO "RELATIONSHIPS"("KEY", "SOURCE", "TARGET", "TYPE") VALUES (30, 'Hephaestus',
'Aphrodite', 'marriedTo');
INSERT INTO "RELATIONSHIPS"("KEY", "SOURCE", "TARGET", "TYPE") VALUES (31, 'Cronus',
'Rhea', 'marriedTo');
INSERT INTO "RELATIONSHIPS"("KEY", "SOURCE", "TARGET", "TYPE") VALUES (32, 'Rhea',
'Cronus', 'marriedTo');
INSERT INTO "RELATIONSHIPS"("KEY", "SOURCE", "TARGET", "TYPE") VALUES (33, 'Uranus',
'Gaia', 'marriedTo');
INSERT INTO "RELATIONSHIPS"("KEY", "SOURCE", "TARGET", "TYPE") VALUES (34, 'Gaia',
'Uranus', 'marriedTo');

CREATE GRAPH WORKSPACE "GRAPH"
  EDGE TABLE "RELATIONSHIPS"
    SOURCE COLUMN "SOURCE"
    TARGET COLUMN "TARGET"
    KEY COLUMN "KEY"
  VERTEX TABLE "MEMBERS"
    KEY COLUMN "NAME"
;

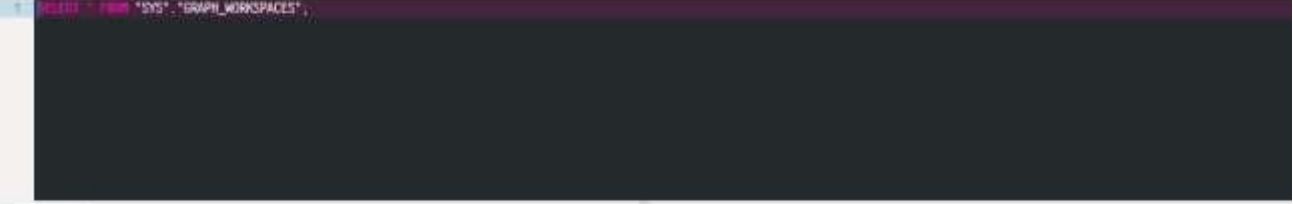
```

To verify that the graph workspace was created successfully, enter the following code and you should see your graph workspace listed:

```
SELECT * FROM "SYS"."GRAPH_WORKSPACES";
```

### Figure 2:

*Listing of All Graph Workspaces in the System*



The screenshot shows a query result table with the following data:

	SCHEMA_NAME	WORKSPACE	CREATE_TIM	USER_NAME	EDGE_SCHE	EDGE_TABLE	EDGE_SOUR	EDGE_TARG	
1	GBL_GRAPH	GBL_GRAPH_DEMO	Mon Sep 19 2016 23:00	SYSTEM	GBL_GRAPH	RELATIONS	SOURCE	TARGET	KE
2	GBL_031	GRAPH	Wed Dec 06 2017 08:50	GBL_031	GBL_031	RELATIONSHIPS	SOURCE	TARGET	KE

The second workspace named "GRAPH" in the screenshot was created by the code used in this assignment.

Next, log onto SAP HANA Graph Viewer and select your graph workspace and you should see the graph taxonomy of your graph as illustrated below (Figure 3):

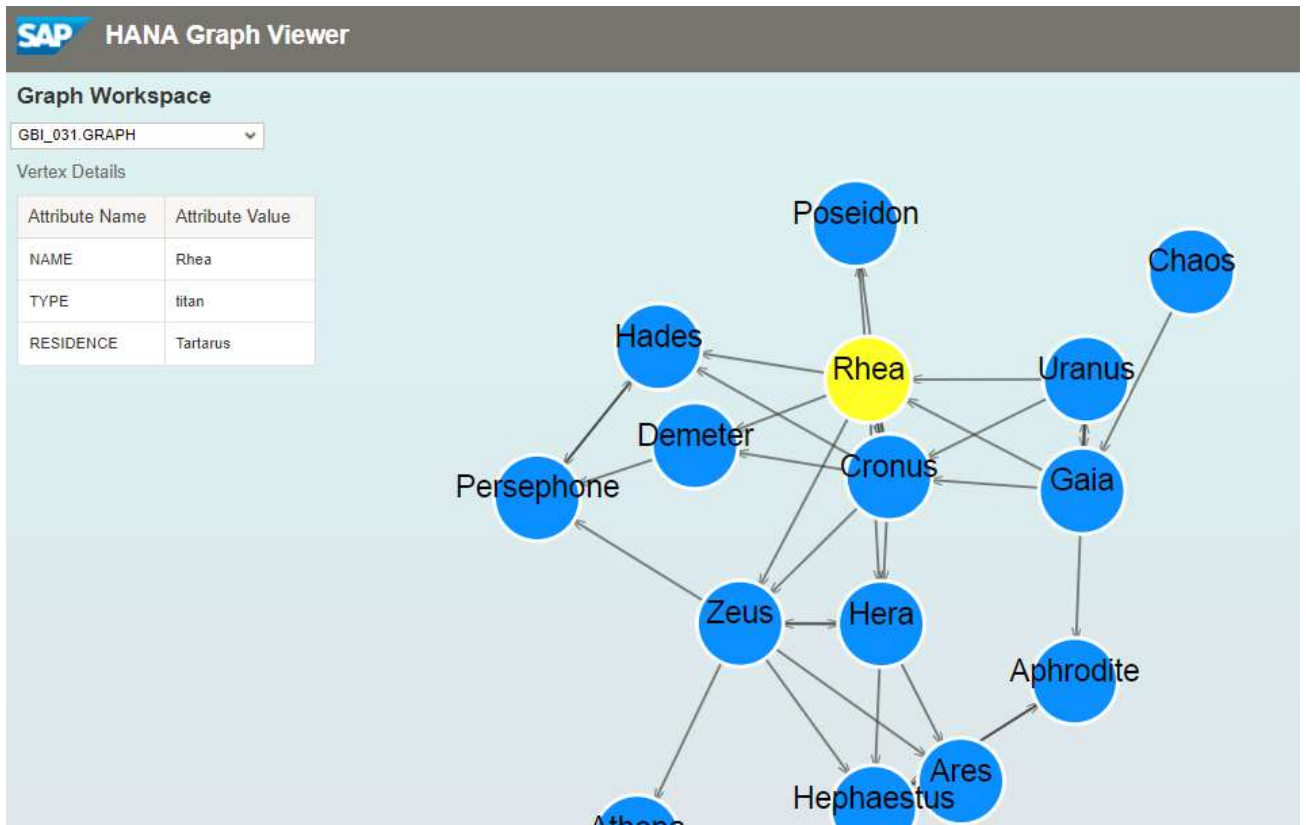
**Figure 3:**  
Graph Taxonomy in the Graph Viewer



**Section II: Modify the graph**

Visualize your graph and see the properties of Rhea as follows (Figure 4):

**Figure 4:**  
Properties of a Vertex



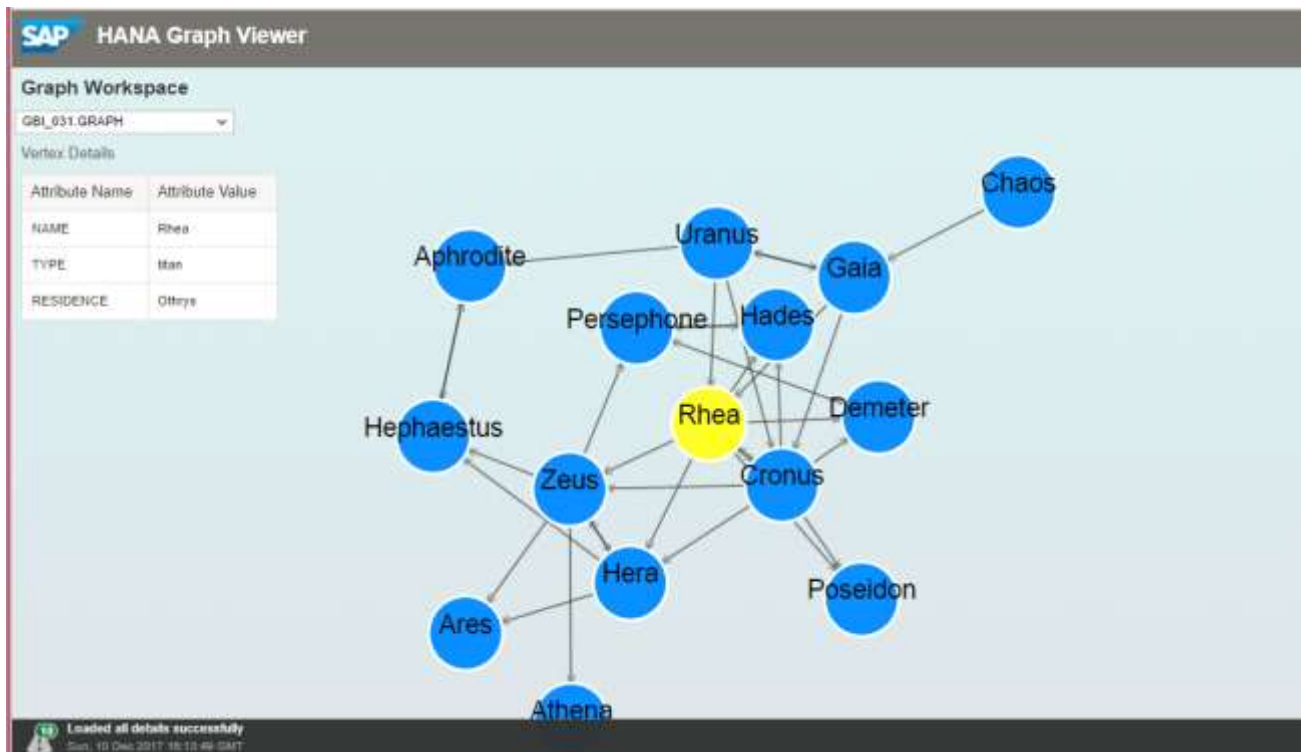


Change her residence to Othrys using the following code:

```
UPDATE "MEMBERS" SET "RESIDENCE" = 'Othrys' WHERE "NAME" = 'Rhea';
```

The next screenshot (Figure 5) shows that her residence has been changed to Othrys.

**Figure 5:**  
*Updated Properties of a Vertex*

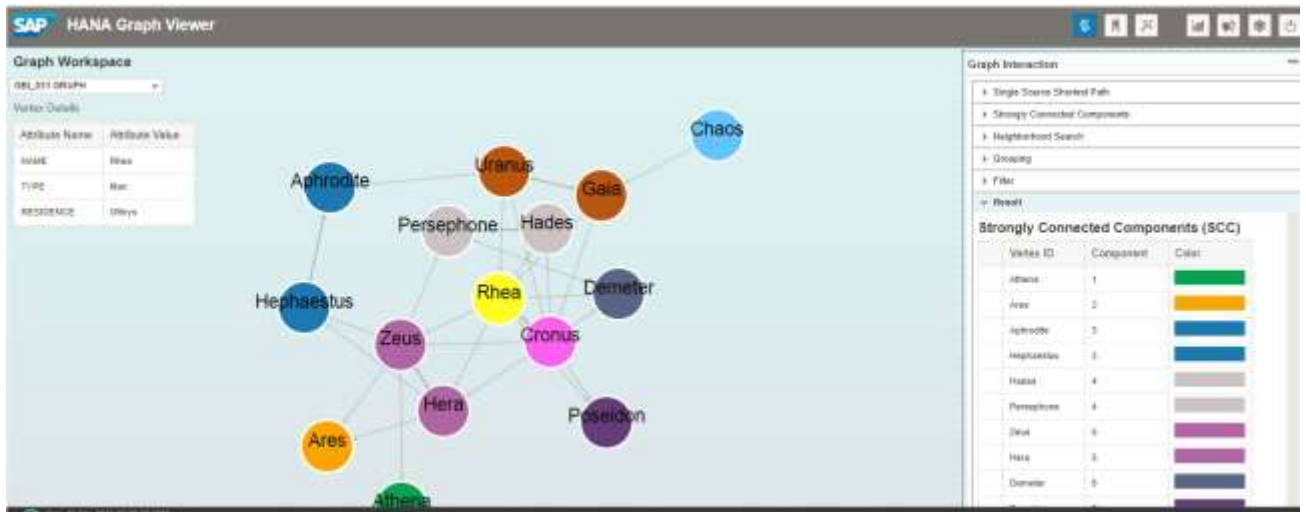


### **Section III: Process the graph**

Graph algorithms include the neighbourhood search, the shortest path and the strongest connected components (SCC). The SCC can be used to partition the graph into subgraphs where every vertex is connected to every other vertex. Executing SCC in the Graph Viewer will result in the output as illustrated below. The component index represents the subgraphs and vertices with the same index number are connected to each other (i.e., married to each other). See if you can tell the number of couples represented in the graph (hint: look for vertices with the same color).

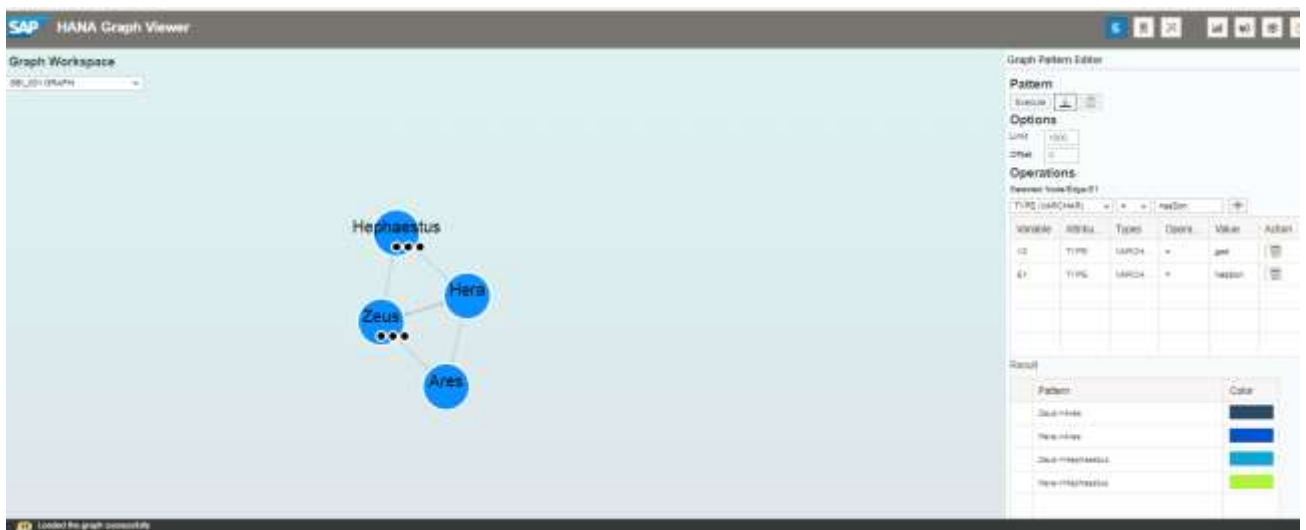
**Figure 6:**  
*Strongly Connected Components*





Pattern matching can be used to find vertices with similar attributes. For example, we can find gods that have sons. The following screenshot shows two gods, Zeus and Hera, with their sons, Hephaestus and Ares.

**Figure 7:**  
*Pattern Matching*



### **Critical thinking questions:**

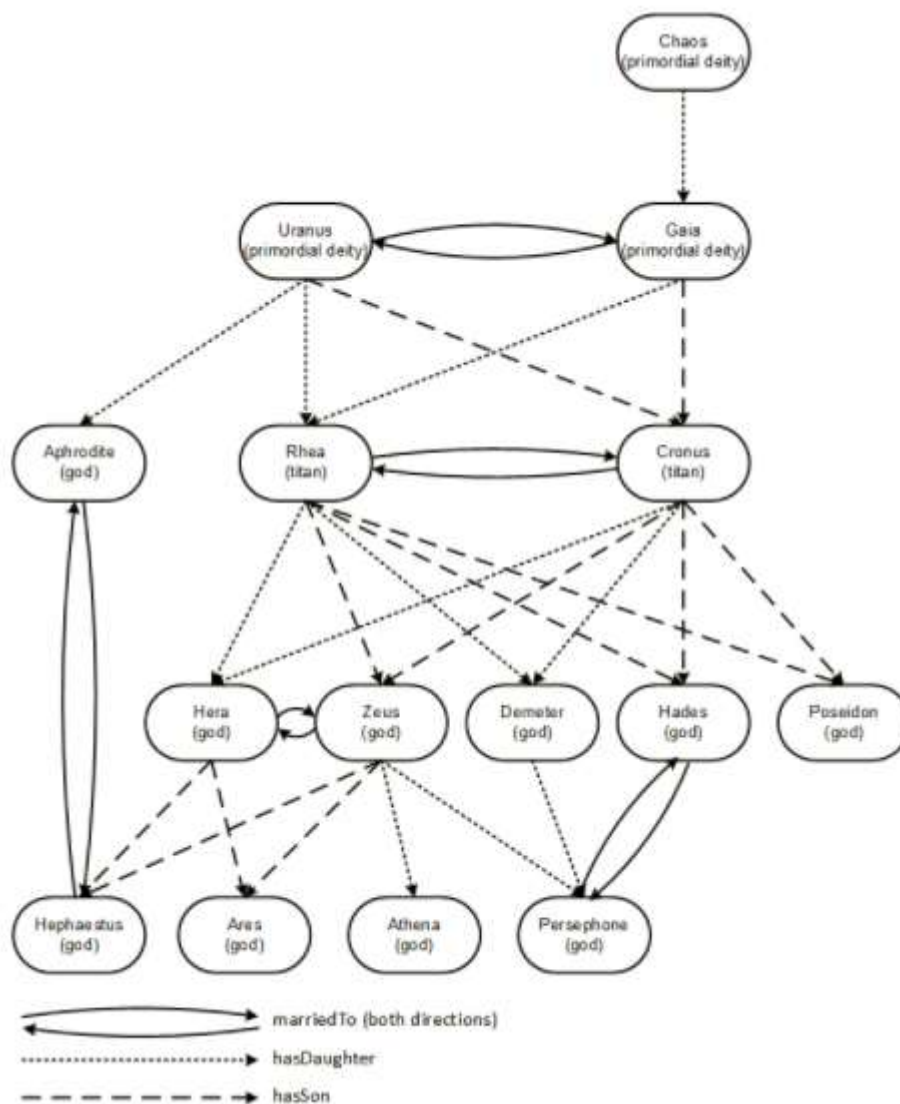
1. Modify the graph by adding or deleting vertices or relationships.
2. Process the graph by executing a graph algorithm (e.g., the shortest path or neighbourhood search). In addition, execute a pattern matching. Explain why this query is important and how it can be used to find answers in other domains (e.g., recommendation systems).

### **Conclusion**

Graph processing represents a critical skill of information systems developers in the age of big data. Knowing how to construct and process a graph enables developers to work

with data integral to wide ranging applications including social networks, logistics, and recommendation systems. While pure graph database systems exist, a hybrid system that combines relational technology with graph processing capability reduces the learning curve. SAP HANA is an in-memory relational database with a built-in graph engine that makes graph processing off of relational tables easy to learn and use. This teaching case details how a small graph database can be created, modified and processed using SAP HANA. Instructors whose schools are members of the SAP University Alliances program can readily incorporate the teaching case into a class on database or emerging technologies.

**Figure 8:**  
*Sample Graph Database*



## References

Elhadi, A. A. E., Maarof, M. A., Barry, B. I. A., & Hamza, H. (2014). Enhancing the Detection of Metamorphic Malware Using Call Graphs, *Computers & Security*, 46, 62-78.

- Hurlburt, G.F., Thiruvathukal, G.K., & Lee, M.R. (2017). The Graph Database: Jack of All Trades or Just Not SQL?, *IT Professional*, 19, 6, 21-25.
- Longbottom, C. (2016). Graph Databases: What Are The Benefits for CIOs?, *Computer Weekly*. Retrieved from <http://www.computerweekly.com/opinion/Graph-databases-What-are-the-benefits-for-CIOs>.
- McKenna, B. (2016). Panama Papers Revealed By Graph Database Visualisation Software, *Computer Weekly*. Retrieved from <http://www.computerweekly.com/news/450280758/Panama-Papers-revealed-by-graph-database-visualisation-software>.
- Mortleman, J. (2016). Graph Databases: Joining the Dots, *Computer Weekly*. Retrieved from <http://www.computerweekly.com/feature/Graph-databases-Joining-the-dots>.
- Robinson, I., Webber, J., & Eifrem, E. (2015). *Graph Databases: New Opportunities for Connected Data*, 2<sup>nd</sup> edition, Sebastopol, CA: O'Reilly.
- Rudolf, M., Paradies, M., Bornhövd, C., & Lehner, W. (2013). The Graph Story of the SAP HANA Database, *BTW*, 403–420.
- Yegulalp, S. (2017). What Is NoSQL? NoSQL Databases Explained, *InfoWorld*. Retrieved from <https://www.infoworld.com/article/3240644/nosql/what-is-nosql-nosql-databases-explained.html>