RESEARCH ARTICLE OPEN ACCESS

Hana Advanced Modeling Graph Processing Techniques

[1]Vnsanyasirao Chittem, [2]Narayana Varma
[1]Senior Lead Consultant Databases, [2]Associate Director – SAP HANA and EDW
[1]Hana Practice Team ,
[1]TekLink Software Ltd, Hyderabad, India
[1]Venkata.nsrao@teklink.com [2]manish.maheshwari@teklink.com

----------**********----------

Abstract:

A graph database is an online database management system with Create, Read, Update and Delete (CRUD) operations working on a graph data model. Unlike other databases, relationships take first priority in graph databases. This means your application doesn't have to infer data connections using things like foreign keys or out-of-band processing, such as MapReduce. The data model for a graph database is also significantly simpler and more expressive than those of relational or other NoSQL databases.

Keywords **—Workspace, Shortest Path Algorithm, Dijkstra shortest path algorithm**

----------**********----------

I. INTRODUCTION

Now-a-days the data stored in a database and which is used for application is huge. SAP HANA Graph is an integral part of SAP HANA core functionality. It expands the SAP HANA platform with native support for graph processing and allows you to execute typical graph operations on the data stored in an SAP HANA system. Enterprises and organizations collect and analyze huge volumes of data that is increasingly connected. Traditional approaches of storing and processing data in relational databases are insufficient when it comes to analyzing "connections" efficiently. In the recent past, graph databases have proven their suitability for novel types of use cases. For example, manufacturing companies

SAP HANA graph provides built-in algorithms and programming models to analyze connected data. As an integral part of the SAP HANA platform, SAP HANA graph combines and bridges the worlds of relational and connected data, thereby reducing system landscape complexity and providing real-time graph analysis. Furthermore, graph analysis can be easily combined with other SAP HANA engines, for example full-text search and geospatial analysis, making SAP HANA an ideal platform for multi-modal applications.

II. CREATING WORKSPACES

First, we must create Hana workspace to use Hana Graph model.

Hana Workspaces can be created using SQL scripts as below.

During the creation of a new graph workspace, SAP HANA Graph assures that all specified tables and columns exist and have supported data types. A newly created graph workspace is valid if the specified columns and tables exist and fulfill the validity requirements:

- vertex key and edge key columns

- Are of the supported key types (TINY INT, SMALL INT, INTEGER, BIG INT, VARCHAR, and NVARCHAR)
- Have NOT NULL flag
- Have UNIQUE flag
- Source and target columns
- Have the same data type as vertex key column
- Have not NOT NULL flag

create your Own schema before creating all the tables to create Graph Work Space.

Create vertex Table Example:

```
Create Column Table Graphvertex(Id Varchar(20),
Primary Key("Id")
);
```

Create Edge Table Example :

```
Create Column Table Graphedge(Id Int Generated By Default As Identity(Start With 1 Increment By 1) Not Null,
Source Varchar(20) Not Null,
Target Varchar(20) Not Null,
PRIMARY KEY("ID')
);
```

Insert some data into Vertex and Edge table before creating workspace and after inserting data into vertex and edge table create workspace using create workspace command in Hana as below.

Create Workspace Example

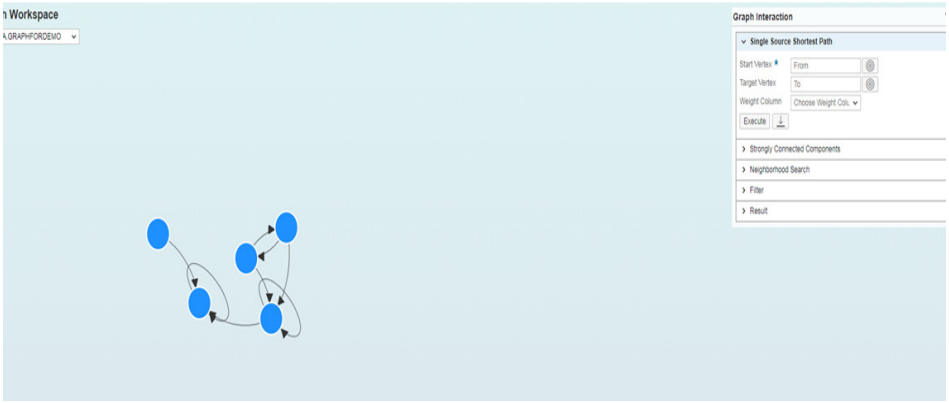
```
Create GRAPH WORKSPACE "GRAPHFORDEMO"
EDGE TABLE "GRAPHEDGE"
SOURCE COLUMN "SOURCE"
TARGET COLUMN "TARGET"
KEY COLUMN "ID"
VERTEX TABLE "GRAPHVERTEX"
KEY COLUMN "ID";
```

Use workspace in web browser.

III. GRAPH ALGORITHMS

Shortest Path ALGORITHMS

After creating workspace and log into web browser , to find out shortest path between nodes different algorithms will play key roles.



Dijkstra's **Shortest Path** algorithm is very similar to Prim's algorithm for minimum spanning tree. Like Prim's MST, we generate a SPT (shortest path tree) with given source as root. We maintain two sets, one set contains vertices included in shortest path tree, other set includes vertices not yet included in shortest path tree. At every step of the algorithm, we find a vertex which is in the other set (set of not yet included) and has a minimum distance from the source.

Below are the detailed steps used in Dijkstra's algorithm to find the shortest path from a single source vertex to all other vertices in the given graph.

Algorithm

1) Create a set sptSet (shortest path tree set) that keeps track of vertices included in shortest path tree, i.e., whose minimum distance from source is calculated and finalized. Initially, this set is empty.

2) Assign a distance value to all vertices in the input graph. Initialize all distance values as INFINITE. Assign distance value as 0 for the source vertex so that it is picked first.

3) While sptSet doesn't include all vertices

….a) Pick a vertex u which is not there in sptSet and has minimum distance value.

….b) Include u to sptSet.

….c) Update distance value of all adjacent vertices of u. To update the distance values, iterate through all adjacent vertices. For every adjacent vertex v, if sum of distance value of u (from source) and weight of edge u-v, is less than the distance value of v, then update the distance value of v.

Single Source Shortest Path Algorithm

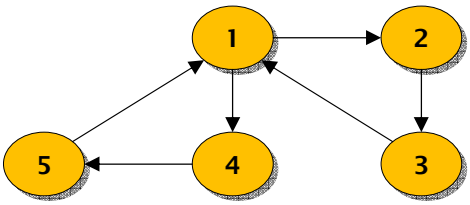
Takes a start vertex, an optional end vertex and an optional edge weight attribute as input parameters

- As output, the algorithm returns the calculated weight („distance) and the path for each target vertex
- Use Cases
  - Trivial: Navigation Systems
    - Navigation system can use all of the attributes that are stored in the relationships in order to better perform (e.g. if two nodes are connected by a street without tarmac, the algorithm should avoid this street)
  - Logistics: Calculating the best route for a delivery
    - The weights between nodes can represent different dimensions: distance, cost, time etc.
    - Depending on the requirements, one or many of the different dimensions can be used to calculate the best route

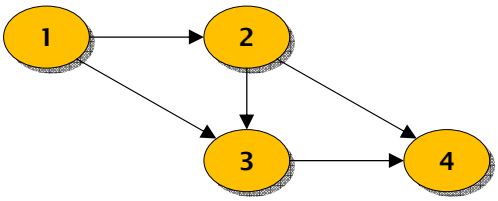
Strongly Connected Components Algorithm

Strongly Connected Components. A directed graph is strongly connected if there is a path between all pairs of vertices. A strongly connected component of a directed graph is a maximal strongly connected subgraph. A directed graph is said to be strongly connected if every vertex is reachable from every other vertex. The strongly connected components of an arbitrary directed graph form a partition into subgraphs that are themselves strongly connected.

Ex:



All vertices can be reached from all vertices:
Graph is strongly connected



No vertex can be reached from vertex 4:
Graph is not strongly connected

Neighborhood Search Algorithm

The algorithm requires a start vertex, a direction (only follow incoming / outgoing edges), a start / end level (distance) as well as filters for vertices and edges

- As output, the algorithm delivers a list of vertices with their depth („distance“from the start vertex)

Example:

Social Network Marketing:

- Find all friends (depth: 0), friends-of-friends (depth:1), friends-of-friends-of-friends (depth:2) and so on
- Find users with a big network which should consist of users that also have big networks, then target these users for marketing campaigns

IV. CALCULATION SCENARIOS IN GRAPH

calculation scenarios for various purposes. Those scenarios are represented as graphs and executed by the calculation engine. In many cases, calculation scenarios are generated implicitly, for example when activating models in HANA studio. BW often makes use of a particular DDL statement (CREATE CALCULATION SCENARIO...) in order to generate them. Upon execution of such a model (scenario) many optimizations are then done inside the calculation engine, eg push down of filters or determining the degree of parallelism.

Starting with HANA SP9, HANA Studio offers the possibility to visualize the graph that represents the calculation scenario.

Example for calculation Scenario as below:

```
CREATE CALCULATION SCENARIO  GET_NEIGHBORHOOD_EXAMPLE" USING '
<?xml version="1.0"?>
<cubeSchema version="2" operation="createCalculationScenario"
defaultLanguage="en">
<calculationScenario                                    schema="GREEK_MYTHOLOGY"
    name="GET_NEIGHBORHOOD_EXAMPLE">
<calculationViews>
<graph name="get_neighborhood_node" defaultViewFlag="true"
schema="GREEK_MYTHOLOGY" workspace="GRAPH" action="GET_NEIGHBORHOOD">
```

```
<expression>
<![CDATA[{
 "parameters": {
 "startVertices": ["Chaos"],
 "direction":"outgoing",
 "minDepth": 0,
 "maxDepth": 2
 }
 }]]>
</expression>
<viewAttributes>
<viewAttribute name="NAME" datatype="string"/>
30 P U B L I C
SAP HANA Graph Reference
Graph Algorithms
<viewAttribute name="DEPTH" datatype="int"/>
</viewAttributes>
</graph>
</calculationViews>
</calculationScenario>
</cubeSchema>
' WITH PARAMETERS ('EXPOSE_NODE'=('get_neighborhood_node',
'GET_NEIGHBORHOOD_EXAMPLE'));
```

The following SQL statement executes the calculation scenario and orders the resulting vertices by depth in reverse order.

```
SELECT * FROM "."GET_NEIGHBORHOOD_EXAMPLE" ORDER BY "DEPTH";
```

The result of this operation is the following vertex table:

NAME	DEPTH
Chaos	0
Gaia	1
Uranus	2
Cronus	2
Rhea	2

The same result can be obtained through the SAP HANA Graph Viewer web tool, but using calculation scenario script through SQL we can get same result.

V. CONCLUSION

According to our observations, the performances of the algorithms are strongly depends on the entropy, information gain and the features of the data sets. There are various work has been done using the Graph Algorithm Some recent improve algorithm reduce problem like replication, handle continuous data,

biased to multi value attribute. This paper provides students and researcher some basic fundamental information about Graph databases in Hana and creating and working with Workspaces.

VI. REFERENCES

[1] https://help.sap.com/doc/21574acf46fe45a8ae9def213f2c4d9e/2.0.00/en-us/sap_hana_graph_reference_en.pdf

[2] https://help.sap.com/viewer/f381aa9c4b99457fb3c6b53a2fd29c02/2.0.01/en-US/3d10ffc605524712adbaf5a00495fc37.html

[3] https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/