**Graph Database Schema**

Our provided Composability Management Framework utilizes a graph database (Janusgraph) to mirror transactions and states from Sunfish. Using this method, fewer RESTful queued interactions need to be made to the Sunfish tree.

Our provided database is designed to scale, using a Cassandra peer-to-peer database. For our proof-of-concept, we constructed our graph database on an Ubuntu OS.

The Graph Resource Storage Table Characteristics are the “top” Characteristics required for each Client request, as programmed by the implementer.

**Janusgraph Resource Storage Table**

|  |  |  |
| --- | --- | --- |
| **Composer Entry Name** | **Value Type** | **Description** |
| ResourceName | String | The Sunfish Resource name |
| ComposerID | Integer | Installed Identification Number |
| ResourceType | String | Sunfish Resource Type |
| ResourceSubType | String | Resource Subtype |
| ResourceActive | Boolean | True or False |
| ResourceAllocated | Boolean | True or False |
| ResourceCharacteristic1 | String | Sunfish Resource Characteristic |
| ResourceCharacteristic2 | String | Sunfish Resource Characteristic |
| ResourceCharacteristic3 | String | Sunfish Resource Characteristic |
| ResourceCharacteristic4 | String | Sunfish Resource Characteristic |
| ResourceCharacteristic5 | String | Sunfish Resource Characteristic |
| ResourceCharacteristic6 | String | Sunfish Resource Characteristic |
| ResourceCharacteristic7 | String | Sunfish Resource Characteristic |
| ResourceCharacteristic8 | String | Sunfish Resource Characteristic |
| ResourceCharacteristic9 | String | Sunfish Resource Characteristic |
| ResourceCharacteristic10 | String | Sunfish Resource Characteristic |
| Message | String | Event Message |
| MessageID | Integer | Event Message ID |
| ProposedResolution | String | Error Event BMC Proposed Resolution Message |
| ResourceEndpointConnectionTypes | String and Comma Separated | Endpoint connection types available to the Resource |
| ResourceEndpointNames | String and Comma Separated | Endpoint connection names |
| ResourceEndpointConnectionBandwidths | Integers and Comma Separated | Bandwidth performance Values for the Connections |
| ResourceLocationPath | String | Reference Resource location path in Sunfish tree |
| AggregatedDevices | String and Comma Separated | Aggregated Devices |
| Tenancy | String | Tenancy |
| SecurityValue | String | Security Property Value |
| SecurityAssociation | String | Security Association |

**Janusgraph Gremlin command line entry commands for the Resource Vertices:**

**Example of Resource Creation:**

g.addV('resource').property('ResourceName',’CXL Memory 1').property('ComposerID',100).property('ResourceType','CXL Memory Module').property(‘ResourceSubType’,’’). property('ResourceActive','False').property('ResourceAllocated','False').property(‘ResourceSpecificCharacteristics’,’Capacity,Speed,Encryption,Granularity,Volatile’).property(‘Capacity’,’32GB’).property(’Speed’,’PCI5 44800’).property(’Encryption’,’True’).property(’Granularity’,’Byte’).property(‘Latency’,’10ns’).property(’Volatile’,’True’).property(‘ResourceCharacteristic7’,’’).property(‘ResourceCharacteristic8’,’’).property(‘ResourceCharacteristic9’,’’).property(‘ResourceCharacteristic10’,’’).property(‘TotalMemory’,’Memory Size’).property(‘MemoryAvailable’,’Memory Size’).property(‘Message’,’Event Message’).property('MessageID',0).property('ProposedResolution','proposed resolution').property('ResourceEndpointConnections','resource connections').property('ResourceEndpointNames','resource endpoint names').property('ResourceEndpointConnectionBandwidths','resource endpoint connection values').property('ResourcePathLocationPath','Sunfish tree path').property('AggregatedDevices','comma separated list').property('tenancy','tenancy').property('security value','security value').property('security association','security association')

**Formal Code Architecture**

g.addV('resource').property('ResourceName',’Resource').property('ComposerID',<id number>).property('ResourceType','Resource Type').property(‘ResourceSubType’,’Resource Sub Type’). property('ResourceActive','Boolean').property('ResourceAllocated','Boolean'). property(‘ResourceCharacteristic1’,’ Resource Characteristic’).property(‘ResourceCharacteristic2’,’ Resource Characteristic’). .property(‘ResourceCharacteristic3’,’ Resource Characteristic’). .property(‘ResourceCharacteristic4’,’ Resource Characteristic’). .property(‘ResourceCharacteristic5’,’ Resource Characteristic’). .property(‘ResourceCharacteristic6’,’ Resource Characteristic’). .property(‘ResourceCharacteristic7’,’ Resource Characteristic’). .property(‘ResourceCharacteristic8’,’ Resource Characteristic’). .property(‘ResourceCharacteristic9’,’ Resource Characteristic’). .property(‘ResourceCharacteristic10’,’ Resource Characteristic’).property(‘TotalMemory’,’Memory Size’).property(‘MemoryAvailable’,’Memory Size’).property(‘Message’,’Event Message’).property('MessageID',0).property('ProposedResolution','proposed resolution').property('ResourceEndpointConnections','resource connections').property('ResourceEndpointNames','resource endpoint names').property('ResourceEndpointConnectionBandwidths','resource endpoint connection values').property('ResourcePathLocationPath','Sunfish tree path').property('AggregatedDevices','comma separated list').property('tenancy','tenancy').property('security value','security value').property('security association','security association')

**g.V().has('ComposerID',100)**

==>v[8288]

**g.V().has('ComposerID',100).values('ResourceName')**

results in:

==>CXL Memory Pool 1

**g.V().has('ComposerID',100).values('ResourceEndpointConnectionBandwidths')**

results in:

==>resource endpoint connection values’

**Janusgraph Interconnect Storage Table**

|  |  |  |
| --- | --- | --- |
| **Composer Entry Name** | **Value Type** | **Description** |
| ResourceName | String | The Sunfish Resource name |
| ComposerID | Integer | Installed Identification Number |
| ConnectionType | String | Sunfish Connection Type |
| ConnectionVersion | String | Sunfish Connection Version |
| LinkSpeed | Long Long | Link Bandwidth |
| Manufacturer | String | Manufacturer Name |
| FECN | Integer | Forward Explicit Congestion Notification |
| BECN | Integer | Backwards Explicit Congestion Notification |
| PerformanceIssues | Boolean | Link has problems |
| BytesTransmitted | LongLong | Counter |
| BytesReceived | LongLong | Counter |
| TransmitDiscards | Integer | Counter |
| TransmitPackets | LongLong | Counter |
| ReceivePackets | LongLong | Counter |
| LinkSpecificRecoveryFlags | Integer | How many times have we recovered from an error? |
| QueueBufferOverrun | Boolean | We ran over the buffer? |
| LinkCharacteristic1 | String | Sunfish Resource Characteristic |
| LinkCharacteristic2 | String | Sunfish Resource Characteristic |
| LinkCharacteristic3 | String | Sunfish Resource Characteristic |
| LinkCharacteristic4 | String | Sunfish Resource Characteristic |
| LinkCharacteristic5 | String | Sunfish Resource Characteristic |
| LinkCharacteristic6 | String | Sunfish Resource Characteristic |
| LinkCharacteristic7 | String | Sunfish Resource Characteristic |
| LinkCharacteristic8 | String | Sunfish Resource Characteristic |
| LinkCharacteristic9 | String | Sunfish Resource Characteristic |
| LinkCharacteristic10 | String | Sunfish Resource Characteristic |
| Message | String | Event Message |
| MessageID | Integer | Event Message ID |
| ProposedResolution | String | Error Event BMC Proposed Resolution Message |
| ResourceLocationPath | String | Reference Resource location path in Sunfish tree |
| Tenancy | String | Tenancy |
| SecurityValue | String | Security Property Value |
| SecurityAssociation | String | Security Association |

g.V(4136).addE('Nic 1 LID Nic 2 LID').to(\_\_.V(4272))

==>e[ao5-36w-i6t-3ao][4136-Nic 1 LID Nic 2 LID->4272]

g.V(4136).addE('Nic 1 LID Nic 2 LID').to(\_\_.V(4272)).property('ConnectionType','RDMA').property('ConnectionVersion','OmniPath').property('LinkSpeed','100Gbps').property('Manufacturer','OmniPath').property('FECN','FECN').property('BECN','BECN').property('BytesSentCounter','Bytes').property('Message','Event Message').property('MessageID',260).property('Proposed Resolution','Proposed Resolution')

g.V(4136).addE('Nic 1 LID Nic 2 LID').to(\_\_.V(4272))..property(‘ResourceName’,’ResourceName’).property(‘ComposerID’,’ComposerID’).property(‘ConnectionType‘,‘ConnectionType’).property(ConnectionVersion,ConnectionVersion’) .property(‘LinkSpeed’,’LinkSpeed’).property(‘Manufacturer’,’Manufacturer’).property(‘FECN’,’FECN’) .property(‘BECN’,’BECN’).property(‘PerformanceIssues’,’PerformanceIssues’) .property(‘BytesTransmitted’,’BytesTransmitted’).property(‘BytesReceived’,‘BytesReceived’).property(TransmitDiscards’,’TransmitDiscards’).property(TransmitPackets’,’TransmitPackets’).property(‘ReceivePackets’,’ReceivePackets’).property(‘LinkSpecificRecoveryFlags’,’LinkSpecificRecoveryFlags’).property(‘QueueBufferOverrun’,’QueueBufferOverrun’).property(‘LinkCharacteristic1’,‘LinkCharacteristic1’).property(‘LinkCharacteristic2’.‘LinkCharacteristic2’).property(‘LinkCharacteristic3,‘LinkCharacteristic3’).property(‘LinkCharacteristic4‘,‘LinkCharacteristic4’).property(‘LinkCharacteristic5‘,’LinkCharacteristic5’).property(‘LinkCharacteristic6’,‘LinkCharacteristic6’).property(‘LinkCharacteristic7’,‘LinkCharacteristic7’).property(‘LinkCharacteristic8’,‘LinkCharacteristic8’).property(‘LinkCharacteristic9’,‘LinkCharacteristic9’).property(‘LinkCharacteristic10’,‘LinkCharacteristic10’).property(‘Message’,’Message’).property(‘MessageID’,’Message ID’).property(‘ProposedResolution’,‘Proposed Resolution‘).property(‘ResourceLocationPath’,’Resource Path’).property(‘Tenancy‘,‘Tenancy’).property(‘SecurityValue’,’Security Value’).property(‘SecurityAssociation’,’Security Association’)

g.E().hasLabel('Nic 1 LID Nic 2 LID')

==>e[a9x-36w-i6t-3ao][4136-Nic 1 LID Nic 2 LID->4272]

g.E().has('MessageID',260)

==>e[emd-36w-i6t-3ao][4136-Nic 1 LID Nic 2 LID->4272

[Using Python - JanusGraph](https://docs.janusgraph.org/interactions/connecting/python/)

Some Gremlin step and predicate names are reserved words in Python. Those names are simply postfixed with \_ in Gremlin-Python, e.g., in() becomes in\_(), not() becomes not\_(), and so on. The other names affected by this are: all, and, as, from, global, is, list, or, and set.

Getting Started with JanusGraph and Gremlin-Python

To get started with Gremlin-Python:

1. Install Gremlin-Python:
2. pip install gremlinpython==3.5.5
3. Create a text file gremlinexample.py and add the following imports to it:
4. from gremlin\_python import statics
5. from gremlin\_python.structure.graph import Graph
6. from gremlin\_python.process.graph\_traversal import \_\_
7. from gremlin\_python.driver.driver\_remote\_connection import DriverRemoteConnection
8. Create a GraphTraversalSource which is the basis for all Gremlin traversals:
9. from gremlin\_python.process.anonymous\_traversal\_source import traversal
10. connection = DriverRemoteConnection('ws://localhost:8182/gremlin', 'g')
11. # The connection should be closed on shut down to close open connections with connection.close()
12. g = traversal().withRemote(connection)
13. # Reuse 'g' across the application
14. Execute a simple traversal:
15. herculesAge = g.V().has('name', 'hercules').values('age').next()
16. print('Hercules is {} years old.'.format(herculesAge))

next() is a terminal step that submits the traversal to the Gremlin Server and returns a single result.