**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.

**Janusgraph Resource Storage Fields**

|  |  |  |
| --- | --- | --- |
| **Composer Entry Name** | **Value Type** | **Description** |
| ResourceName | String | The Sunfish Resource name |
| ComposerID | Integer | Installed Identification Number |
| ResourceType | String | Sunfish Resource Type |
| ResourceAllocated | Boolean | True or False |
| TotalMemory | Long Long | Total Memory in the Resource |
| MemoryAvailable | Long Long | Available Memory in the Resource |
| 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:**

g.addV('resource').property('ResourceName','CXL Memory Pool 1').property('ComposerID',100).property('ResourceType','Resource Type').property('ResourceAllocated','Boolean').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 Fields**

|  |  |  |
| --- | --- | --- |
| **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 |
| BytesSent | Integer | Counter |
| 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.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]