Deep dive on Amazon Neptune

Brad Bebee
Leader, Neptune
Amazon Web Services

Karthik Bharathy
Product Leader, Neptune
Amazon Web Services

Ora Lassila
Principal Graph Technologist, Neptune
Amazon Web Services
Related breakouts

ADM203 Reimagining advertising analytics & identity resolution at scale
ADM301 Best practices for identity resolution with Amazon Neptune
AIM341 Deep learning on graphs
DAT220 Real-world customer use cases with Amazon Neptune
DAT329 Building your first graph application with Amazon Neptune
DAT341 Best practices for graph data modeling and Amazon Neptune
DAT347 Neptune best practices: How to optimize your graph queries
DVC06 Use Neptune to discover where & when events can impact local businesses
GAM303 How Call of Duty uses ML to personalize player engagement
MOB318 AWS AppSync does that?: Support for alternative data
Graphs are all around us
Graph use cases

- Social networking
- Recommendations
- Knowledge graphs
- Fraud detection
- Life Sciences
- Network & IT operations

Connected data

- Navigate (variably) connected structure
- Filter or compute a result on the basis of the strength, weight, or quality of relationships
Leading graph models and frameworks

- Property graph
  - Open source Apache TinkerPop
  - Gremlin traversal language
- Resource Description Framework (RDF)
  - W3C standard
  - SPARQL Query Language

Many customers want both
Amazon Neptune high-level architecture

AMAZON NEPTUNE
High Performance Graph Engine (Durable, ACID with Immediate Consistency)

- Social Networking
- Fraud Detection
- Recommendations
- Knowledge Graphs
- Life Sciences
- Network & IT Ops

TinkerPop / Gremlin
RDF / SPARQL

Cloud-native Storage Service
- Multi-AZ HA
- Read Replicas
- Encryption-at-rest

Bulk load from Amazon S3
Database management
Distributed storage architecture

- Performance, availability, durability
- Scale-out replica architecture
- Shared storage volume with 10-GB segments striped across hundreds of nodes
- Data are replicated 6 times across 3 AZs
- Hotspot rebalance, fast database recovery
- Log applicator embedded in storage layer

Delivered as a managed service
Read replicas and high availability

Performance
• Applications can scale out read traffic across up to 15 read replicas

Low replica lag
• Typically <10 ms
• Master ships redo logs to replica
• Cached pages have redo applied
• Uncached pages from shared storage

Availability
• Failing database nodes are automatically detected and replaced
• If primary fails, a replica replaces it (failover time typically <60 seconds)
• Primary upgrade by forced failover

Amazon Neptune read scaling

Neptune primary
70% Write
30% Read

Neptune replica
100% New reads

Shared Multi-AZ storage
**Air routes dataset**

- Models the world’s airline route network
- Queries operating over the airport connectivity graph
- Sample queries
  - Given
    - Source and target airport
  - Find
    - All one-stop connections

https://github.com/krlawrence/graph/tree/master/sample-data
"Find all of the airport codes for one-stop connections from SEA to FRA"

```java
# Gremlin
g.V() // start out with all vertices
.has('code','SEA') // select vertices having code = 'SEA'
.out('route') // follow 'route' edge
.as('via') // save node in variable 'via'
.out('route') // follow 'route edge again
.has('code','FRA') // assert we ended up in FRA
.select('via') // jump back to the via airport
.values('code') // select airport code
```
Neptune graph data model

Subject | Predicate | Object | Graph
---|---|---|---
22 | code | "SEA" | default
22 | route | 105 | e1
22 | route | 130 | e2
22 | route | 122 | e3
105 | code | "HND" | default
105 | route | 52 | e4
130 | code | "LAS" | default
130 | route | 52 | e5
122 | code | "ICN" | default
122 | route | 52 | e6
52 | code | "FRA" | default

"Find all of the airport codes for one-stop connections from SEA to FRA"

# SPARQL

PREFIX airport: <http://kelvinlawrence.net/air-routes/resource/airport/>
PREFIX edge: <http://kelvinlawrence.net/air-routes/objectProperty/> 
PREFIX prop: <http://kelvinlawrence.net/air-routes/datatypeProperty/>

SELECT ?viaCode WHERE {
  airport:SEA edge:route ?via .
}
Neptune graph data model – indices

<table>
<thead>
<tr>
<th>Subject</th>
<th>Predicate</th>
<th>Object</th>
<th>Graph</th>
</tr>
</thead>
</table>

SPOG – Uses a key composed of Subject + Predicate + Object + Graph

Efficient lookup whenever a prefix of the positions, such as the vertex (subject) or vertex and property identifier, is bound: *Find airport:SEA (22) with code “SEA”*

POGS – Uses a key composed of Predicate + Object + Graph + Subject

Efficient access when only the edge or property label stored in P position is bound: *What nodes have code “SEA”?*

GPSO – Uses a key composed of Graph + Predicate + Subject + Object

Efficient access with the graph (or edge ID) and a property identifier is bound: *What edges are have routes to “FRA”?*

Making the most of your graph queries

https://docs.aws.amazon.com/neptune/latest/userguide/gremlin-explain.html

Analyzing Neptune Query Execution Using Gremlin Explain

Amazon Neptune has added a Gremlin feature named `explain`. This feature is a self-service tool for understanding the execution approach taken by the Neptune engine. You invoke it by adding an `explain` parameter to an HTTP call that submits a Gremlin query.

The `explain` feature provides information about the logical structure of query execution plans. You can use this information to identify potential evaluation and execution bottlenecks. You can then use `query hints` to improve your query execution plans.

Topics

- Understanding How Gremlin Queries Work in Neptune
- Using the Gremlin Explain API in Neptune
- Gremlin Profile API in Neptune
Making the most of your graph queries

https://docs.aws.amazon.com/neptune/latest/userguide/sparql-explain.html

Analyzing Neptune Query Execution Using SPARQL Explain

Amazon Neptune has added a SPARQL feature named explain. This feature is a self-service tool for understanding the execution approach taken by the Neptune engine. You invoke it by adding an explain parameter to an HTTP call that submits a SPARQL query.

The explain feature provides information about the logical structure of query execution plans. You can use this information to identify potential evaluation and execution bottlenecks. You can then use query hints to improve your query execution plans.

Topics

- How the SPARQL Query Engine Works in Neptune
- How to Use SPARQL Explain to Analyze Neptune Query Execution
- Examples of Invoking SPARQL Explain in Neptune
- Neptune Explain Operators
- Limitations of SPARQL Explain in Neptune

AWS Database Blog

Using SPARQL explain to understand query execution in Amazon Neptune

by Taylor Riggan and Michael Schmidt | on 17 SEP 2019 | in Amazon Neptune | Permalink | Comments | Share

Customers continue to want greater visibility and control over the services they use within AWS. When it comes to our database services, customer requests typically revolve around providing greater insights into the query optimization and processing within a given database. Database developers and administrators are mostly already familiar with the idea and use of database query execution plans. Motivated by customer discussions, Amazon Neptune has now added the addition of a SPARQL query explain feature.

Amazon Neptune is a fast, reliable, fully managed graph database, optimized for storing and querying highly connected data. It is ideal for online applications that rely on navigating and leveraging connections in their data.

Amazon Neptune supports W3C Resource Description Framework (RDF) graphs that can be queried using the SPARQL query language. It also supports Apache TinkerPop property graphs that can be queried using the Gremlin graph traversal and query language.

Product themes

- Developer experience
- Performance and enterprise features
- Compliance and regional expansion
Neptune general availability

- Announced on 5/30/2018
- 16 AWS Regions: US East (N. Virginia, Ohio), US West (Oregon), Canada (Central) Europe (Ireland, London, Frankfurt, Stockholm), Middle East (Bahrain), Asia Pacific (Mumbai, Singapore, Sydney, Seoul, Tokyo), GovCloud (East, West)
- Encryption-at-rest with AWS Key Management Service (AWS KMS)
- Encryption-in-transit with TLS 1.2 client connections
- ISO, HIPAA, SOC, PCI/DSS compliance certifications
## New features in Amazon Neptune

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Neptune Streams</strong></td>
<td>• Generate a complete sequence of change-log entries, which record every change made to graph</td>
</tr>
<tr>
<td><strong>Transaction semantics</strong></td>
<td>• Formalized semantics to help you avoid data anomalies</td>
</tr>
<tr>
<td><strong>Gremlin/SPARQL Explain</strong></td>
<td>• Gain insights into the query plan and evaluation order</td>
</tr>
<tr>
<td><strong>SPARQL 1.1 Federated Query</strong></td>
<td>• Use SPARQL to express queries across diverse data sources</td>
</tr>
<tr>
<td><strong>Gremlin sessions</strong></td>
<td>• Client starts session transaction. All queries run during the session are committed only after connection is closed</td>
</tr>
<tr>
<td><strong>Database cloning</strong></td>
<td>• Create multiple clones of a DB cluster using copy-on-write semantics</td>
</tr>
<tr>
<td><strong>Elasticsearch integration</strong></td>
<td>• Full-text search using Elasticsearch with graph data in Neptune</td>
</tr>
<tr>
<td><strong>Neptune Workbench</strong></td>
<td>• In-console notebook experience to query your graph</td>
</tr>
</tbody>
</table>
SPARQL 1.1 Federated Query

Execute a portion of the query against a remote SPARQL endpoint and compose the final query result

Augment local data with external data

Federate Neptune clusters protected by the AWS Identity and Access Management (IAM) service
Neptune Streams

Easy way to capture changes in your graph; log changes to your graph as they happen for both Property Graph and RDF

```
{"lastEventId": {
  "commitNum": 1,
  "opNum": 1
},
"lastTrxTimestamp": 1571059225504,
"format": "GREMLIN_JSON",
"records": [
  {
    "eventId": {
      "commitNum": 1,
      "opNum": 1
    },
    "data": {
      "id": "feb6e536-a9c4-d9fd-57aa-dbb8f94328d6",
      "type": "vl",
      "key": "label",
      "value": {
        "value": "person",
        "dataType": "String"
      }
    },
    "op": "ADD"
  },
  {
    "eventId": {
      "commitNum": 1,
      "opNum": 1
    },
    "data": {
      "id": "feb6e536-a9c4-d9fd-57aa-dbb8f94328d6",
      "type": "vl",
      "key": "label",
      "value": {
        "value": "person",
        "dataType": "String"
      }
    },
    "op": "ADD"
  },
  {
    "eventId": {
      "commitNum": 1,
      "opNum": 1
    },
    "data": {
      "id": "feb6e536-a9c4-d9fd-57aa-dbb8f94328d6",
      "type": "vl",
      "key": "label",
      "value": {
        "value": "person",
        "dataType": "String"
      }
    },
    "op": "ADD"
  }
],
"totalRecords": 1
```
Capture graph changes using Neptune Streams

Neptune and Elasticsearch

- Full-text search queries
- Match, fuzzy, prefix, query_string, etc.
- SPARQL and Gremlin
Neptune Elasticsearch integration

Leverage Elasticsearch for graph data in Amazon Neptune

**SPARQL**

```sparql
SELECT * WHERE {
  SERVICE <http://aws.amazon.com/neptune/vocab/v01/fts#search> {
    ?desc fts:query "regional" .
    ?desc fts:maxResults 100 .
  }
}
```

**Gremlin**

```java
g.withSideEffect("Neptune#fts.endpoint", "https://....amazonaws.com").v().has('desc','Neptune#fts regional').local(values('code','desc').fold()).limit(100)
```

```java
ElasticSearchParams.builder()
  .withQuery("regional municipal")
  .withMaxResults(1)
  .returnSubjectTo("desc")
  .build()
```

```java
==> [HYA, Barnstable Municipal Boardman Polando Field]
==> [SPS, Sheppard Air Force Base–Wichita Falls Municipal Airport]
==> [ABR, Aberdeen Regional Airport]
==> [SLK, Adirondack Regional Airport]
==> [BFD, Bradford Regional Airport]
```
Export an existing Neptune cluster to Elasticsearch


Export Neptune to Elasticsearch

The Neptune Full-text Search CloudFormation templates provide a mechanism for indexing all new data that is added to an Amazon Neptune database in Elasticsearch. However, there are situations in which you may want to index existing data in a Neptune database prior to enabling the full-text search integration.

This solution allows you to index existing data in an Amazon Neptune database in Elasticsearch.

Once you have populated Elasticsearch with your existing Neptune data, you can remove this solution from your account.

Prerequisites

Before provisioning the solution ensure the following conditions are met:

- You have an existing Neptune cluster and an existing Elasticsearch cluster in the same VPC
- Elasticsearch is version 7.1 or above
- You have at least one subnet with a route to the internet:
  - Either, a subnet with the Auto-assign public IPv4 address set to Yes, a route table with a route destination of 0.0.0.0/0, and an internet gateway set to Target (for example, igw-1a2b3c4d).
  - Or, a subnet with the Auto-assign public IPv4 address set to No, a route table with a route destination of 0.0.0.0/0, and a NAT gateway set to Target (for example, nat-12345678901234567). For more details, see Routing.
- You have VPC security groups that can be used to access your Neptune and Elasticsearch clusters.
Announcing Neptune Workbench

Query Amazon Neptune easily using Jupyter notebooks
Try out checking the status of your cluster

The \$status magic can be used to check for the health of your cluster, as well as retrieve additional information such as the engine version.

```
In [1]: %status

```

Try issuing some SPARQL queries

You can issue a SPARQL query by using the \$sparql cell magic. Let's add a few items to the graph and then retrieve them:

```
In [1]: %sparql

```
Resources
Documentation

Start with the ‘What is Neptune?’ and ‘Neptune Overview’ sections
Explore use cases using Neptune


NotebookContentS3Locations: s3://ianrob-examples/notebooks/*
Learn Gremlin

http://kelvinlawrence.net/book/Gremlin-Graph-Guide.html
https://github.com/krlawrence/graph
Learn SPARQL

https://logd.tw.rpi.edu/tutorial/a_crash_course_on_sparql
https://www.w3.org/TR/sparql11-query/
Reference architectures

https://github.com/aws-samples/aws-dbs-refarch-graph/
Samples

https://github.com/aws-samples/amazon-neptune-samples
Use cases, videos, blog posts, and code

https://aws.amazon.com/neptune/developer-resources/
Learn databases with AWS Training and Certification

Resources created by the experts at AWS to help you build and validate database skills

25+ free digital training courses cover topics and services related to databases, including:

- Amazon Aurora
- Amazon Neptune
- Amazon DocumentDB
- Amazon DynamoDB
- Amazon ElastiCache
- Amazon Redshift
- Amazon RDS

Validate expertise with the new AWS Certified Database - Specialty beta exam

Visit aws.training
Thank you!

Brad Bebee
beebs@amazon.com
Please complete the session survey in the mobile app.