
AWS
re:Invent

DAT309

Deep dive on Amazon Neptune

Brad Bebee

GM, Amazon Neptune

AWS

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Agenda

Graphs help you build awesome new things

What’s inside of Neptune?

How does Neptune store and query the graph?

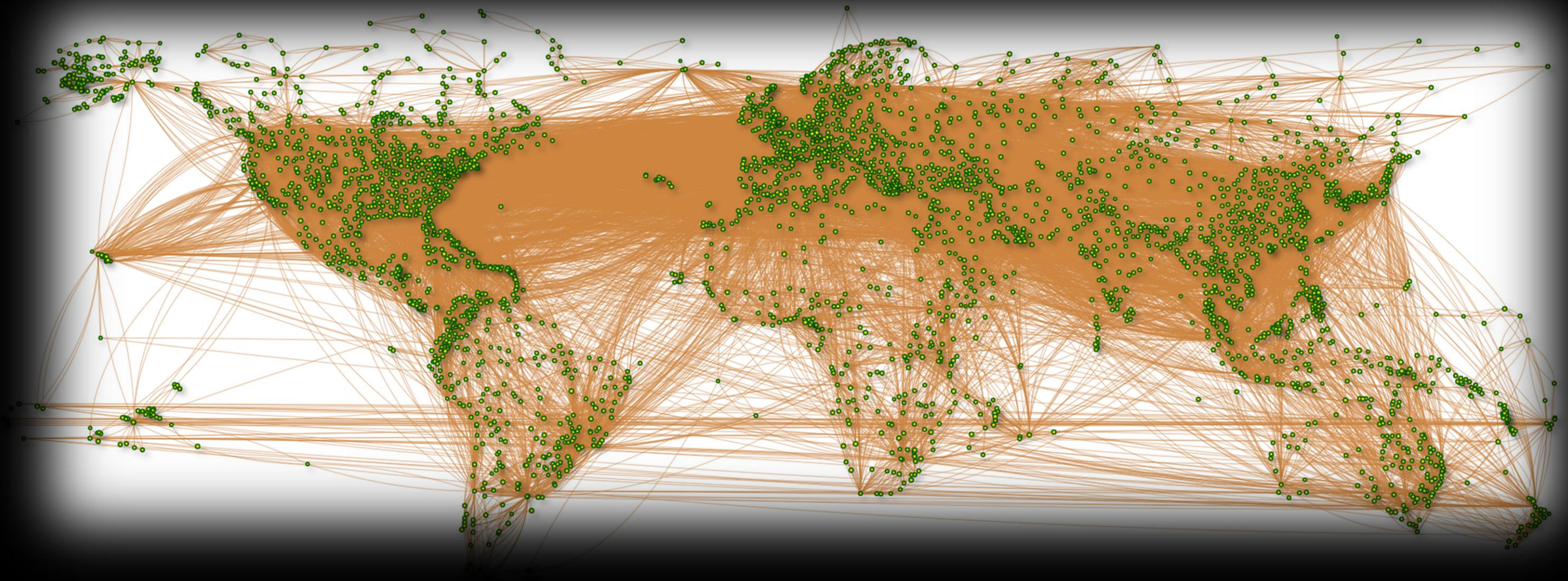
Tips to get the most of your graph application

Did I mention that graphs are awesome? :-)

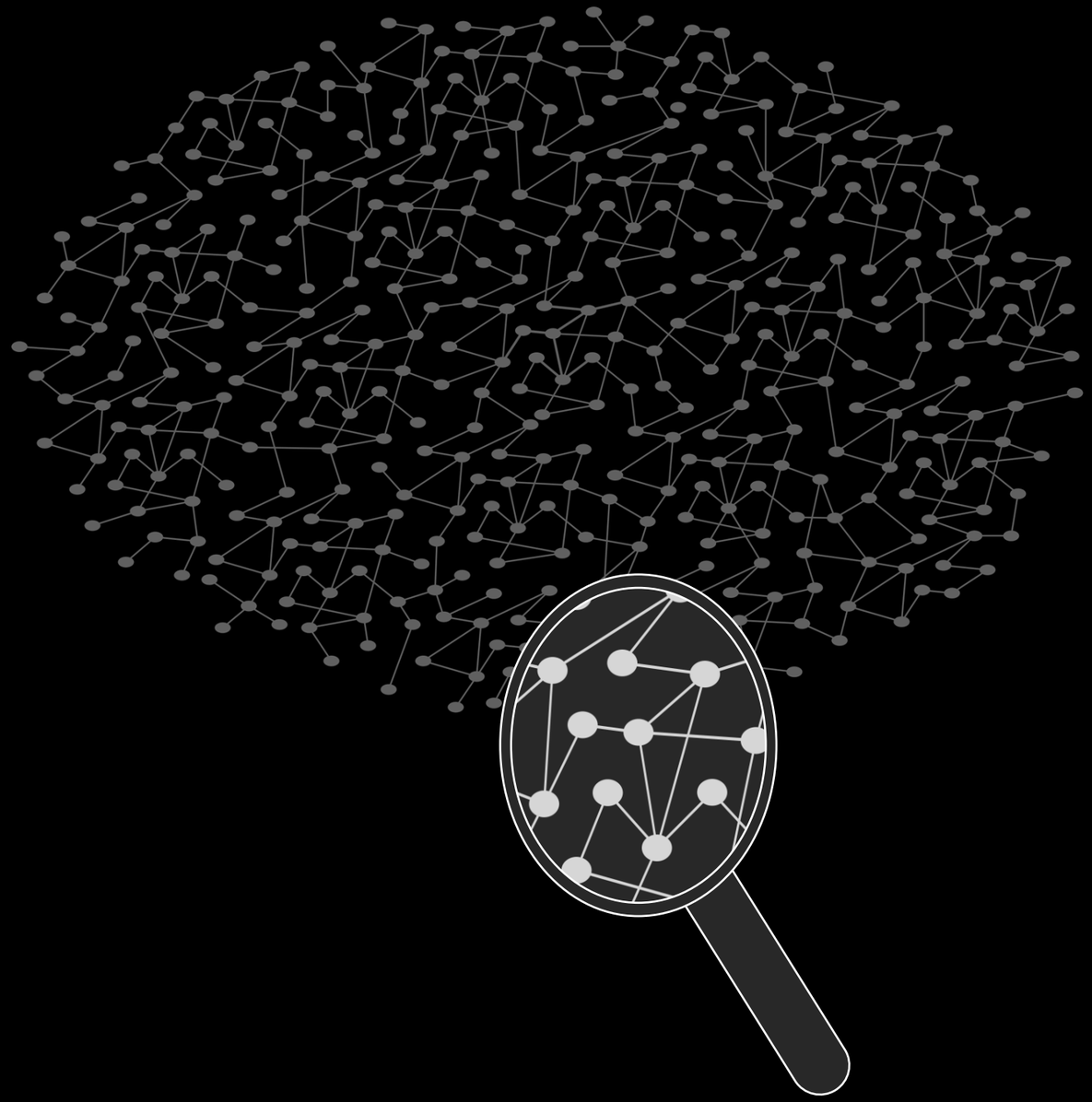


© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Graphs are all around us



When should I use a graph (database)?



Model data based on relationships

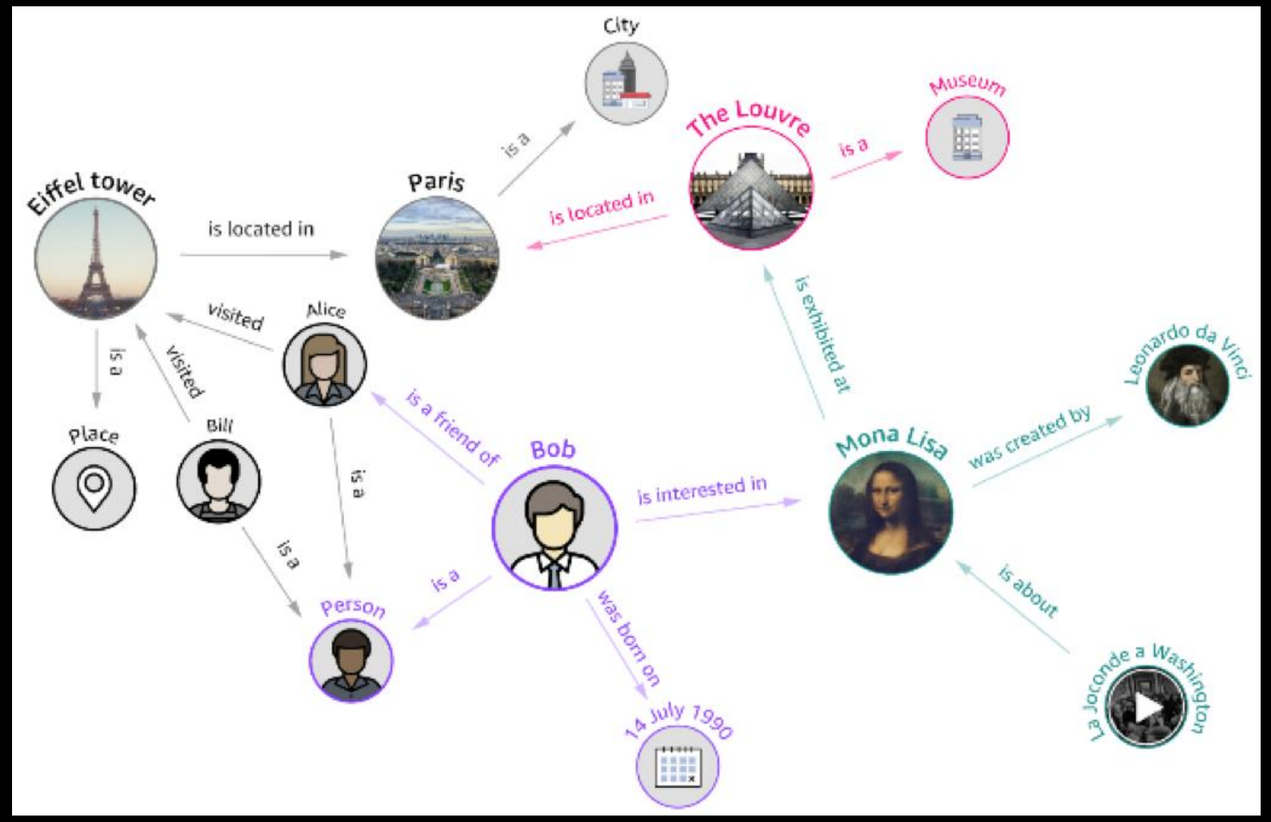
Explore connections and patterns in connected data

Navigate (variably) connected structure

Filter or compute a result on the basis of the strength, weight, or quality of relationships

Build a knowledge graph

UNDERSTANDING THE WHO, WHAT, WHEN, AND WHERE



Benefits

1. Link disparate data sources

Link disparate and heterogeneous data sources together to discover hidden connections

2. Improved search results

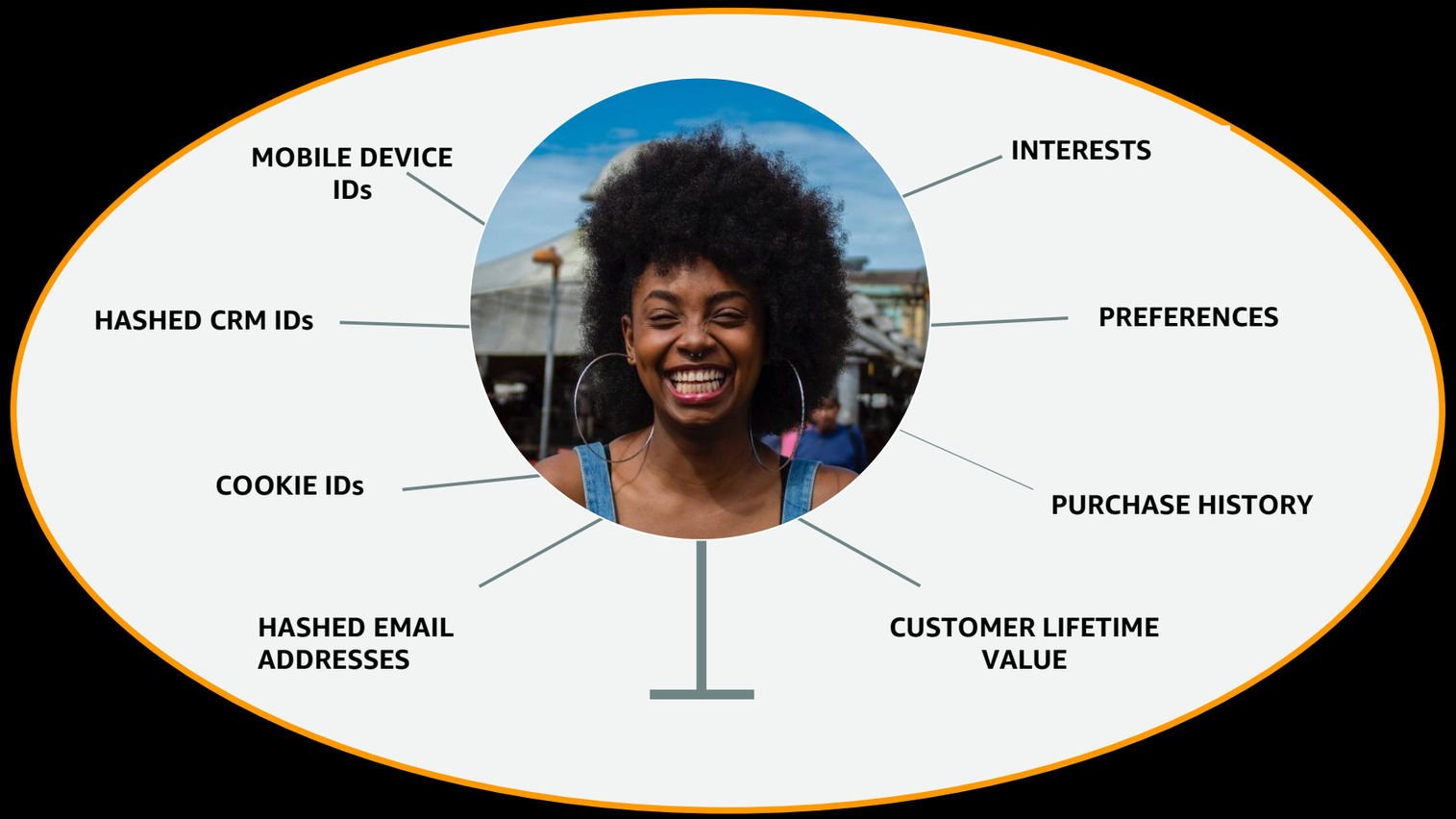
Increase productivity by making data easily accessible through improved search relevance

3. Augment machine learning and artificial intelligence

Improve the efficiency and effectiveness of machine learning models by providing context and augmentation with related content

Build an identity graph

CREATING A UNIFIED 360º VIEW OF THE CUSTOMER



Benefits

1. Persistent IDs across devices and channels

Persistent identifier to link all related devices and IDs, enabling unified profile creation, targeting, and personalization

2. Audience building and segmentation

Create audiences based on similar interests, preferences, and purchases

3. Cross-device frequency capping

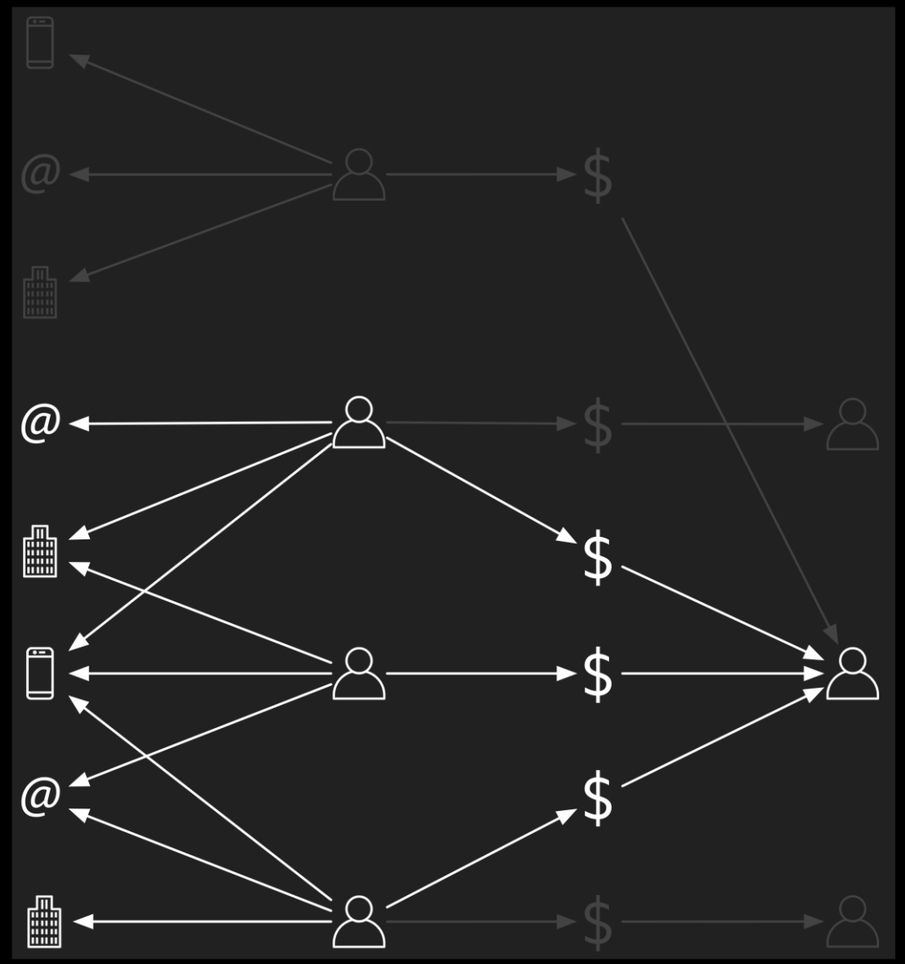
Improve customer experience with better frequency capping across devices and browsers for your audiences

4. Analyze customer journey

Analyze end-to-end customer behavior, and track path to purchase and attribution

Build a fraud graph

DETECTING FRAUD AS IT HAPPENS



Benefits

1. Prevent fraud in real time

Locate and prevent fraudulent patterns of transactions as they are occurring, not after

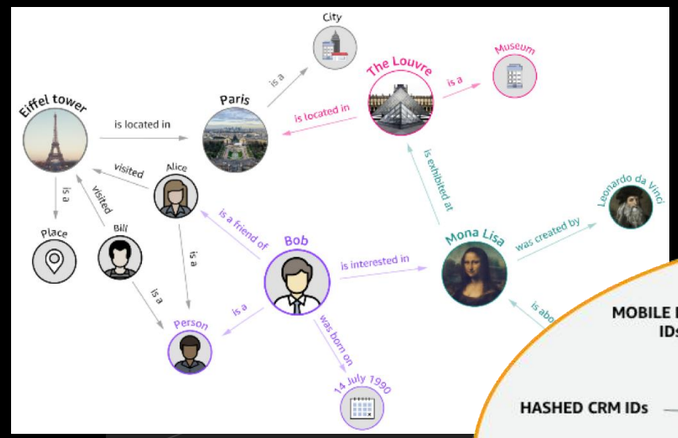
2. Unique insight

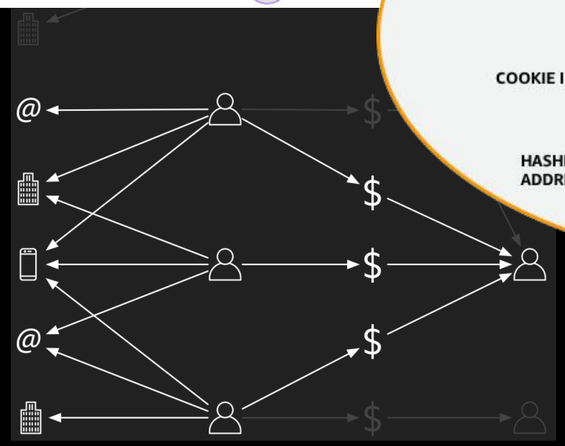
Derive unique insights commonly used to identify fraud (e.g., communities, connected components) using graph algorithms

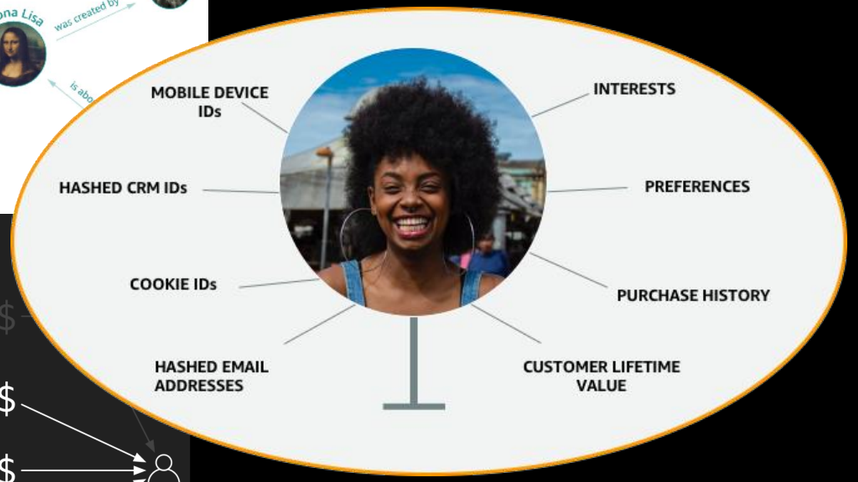
3. Adapt to change

Adapt to the changing nature of fraud quickly and with minimal effort using the flexible nature of connected data

Thousands of customers are in production with knowledge graphs, fraud graphs, and identity graphs







amazon alexa · zeta · THOMSON REUTERS

Pearson · COX AUTOMOTIVE · PaySense

SIEMENS · Rappi · audible FOR BUSINESS

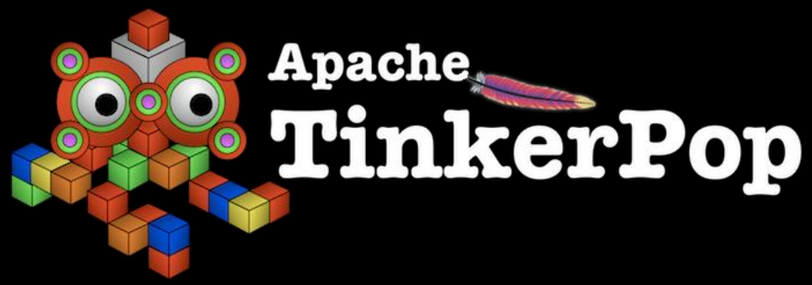
Example Amazon Neptune public reference customers using a knowledge graph, fraud graph, or identity graph.

Leading graph models and frameworks

Property graph

Open source Apache TinkerPop

Gremlin traversal language



Resource Description Framework (RDF)

W3C standard

SPARQL query language



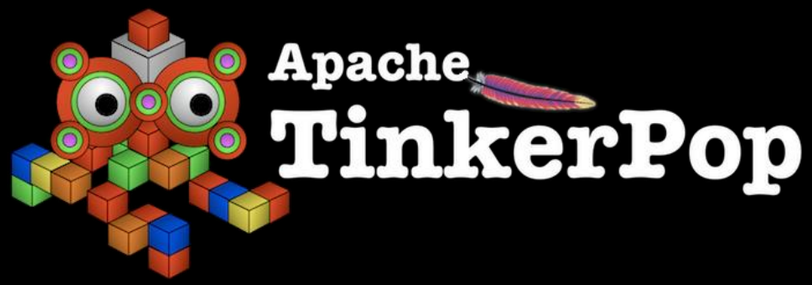


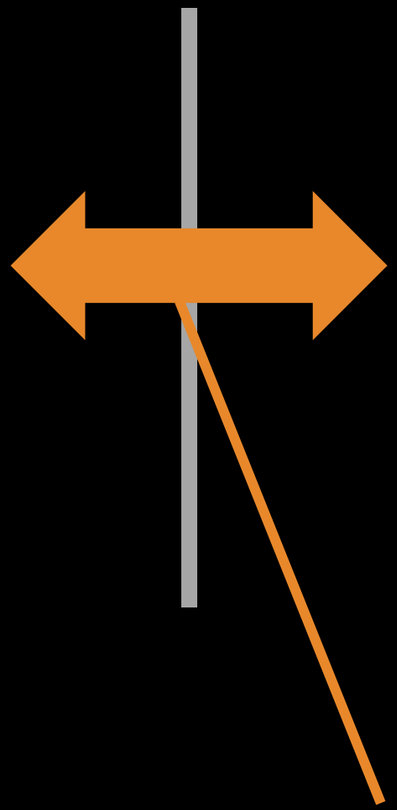
Leading graph models and frameworks

Property graph

Open source Apache TinkerPop

Gremlin traversal language





Resource Description Framework (RDF)

W3C standard

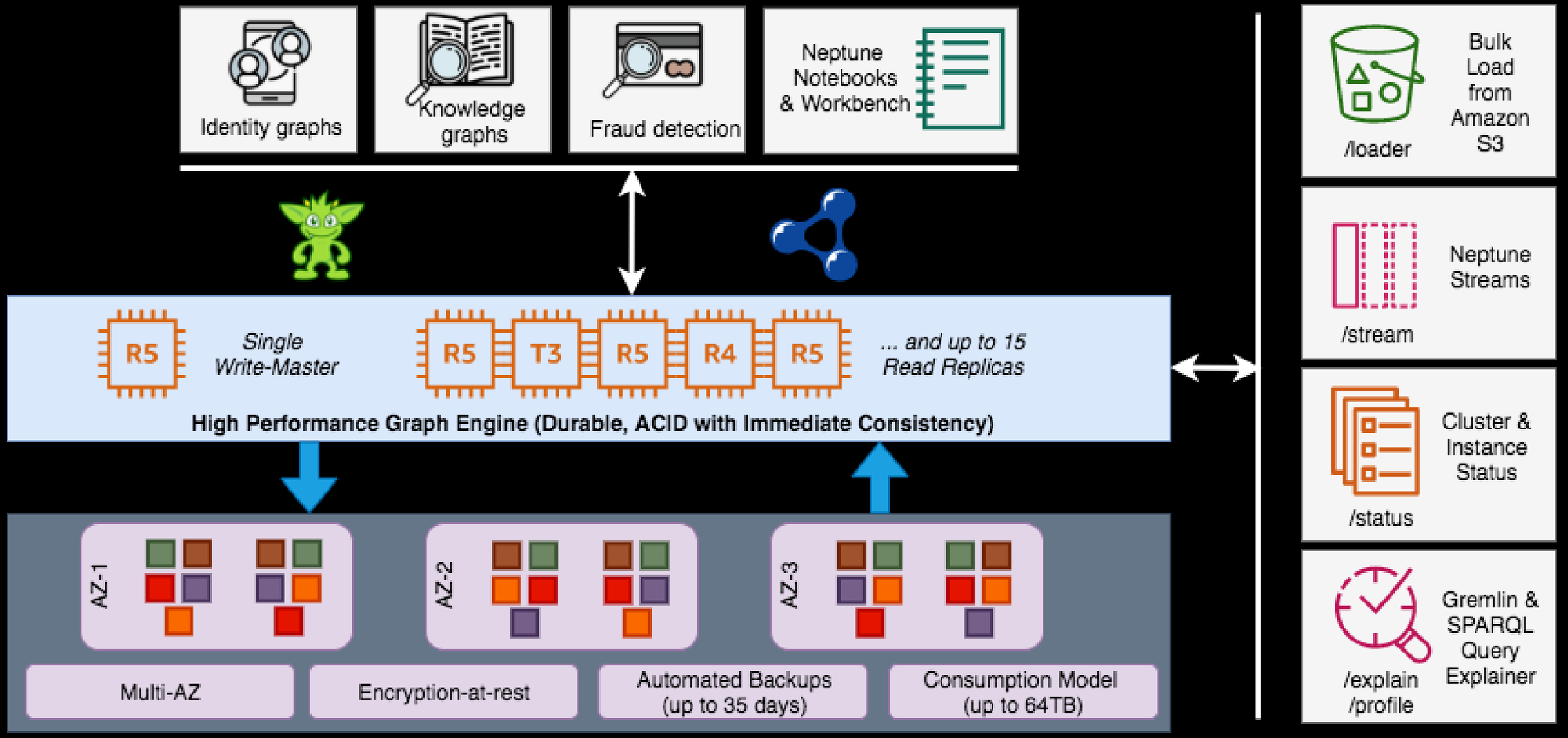
SPARQL query language





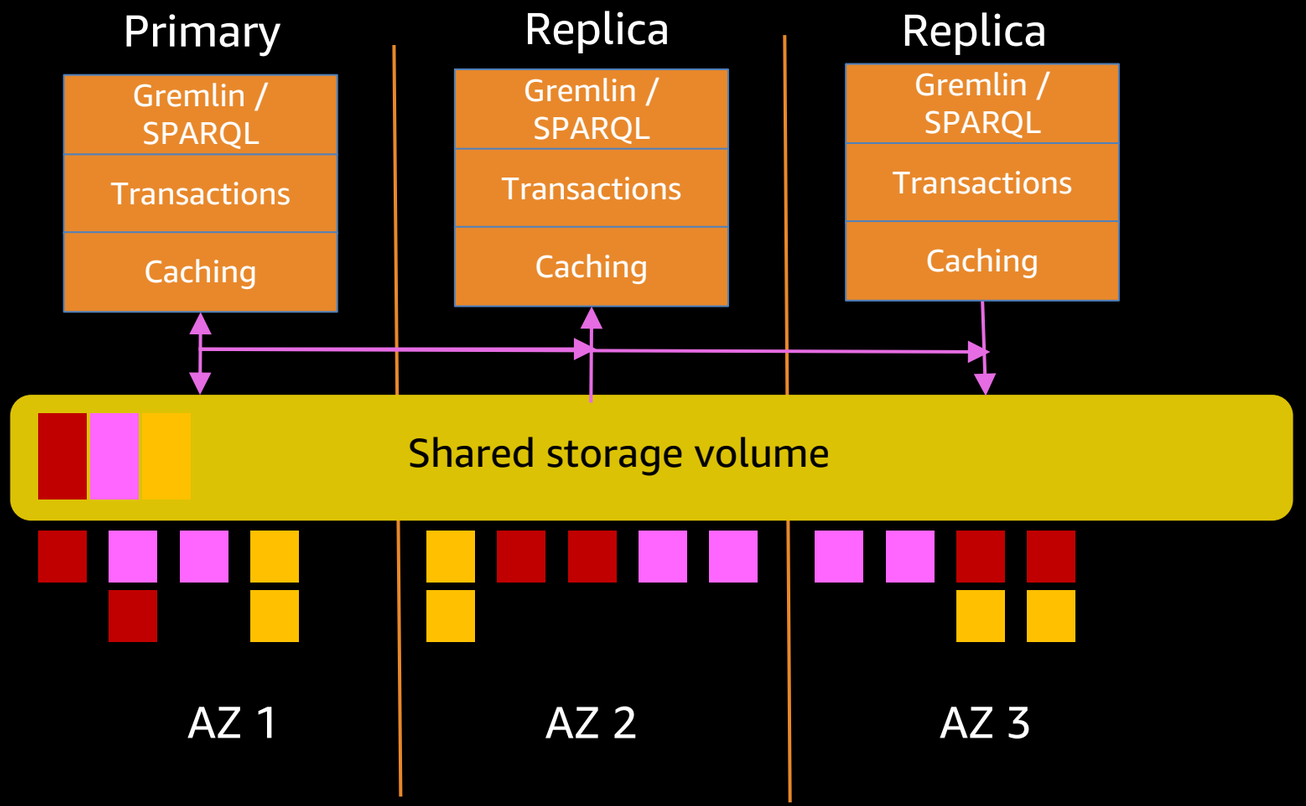
Many customers want both

What's inside of Amazon Neptune?



Distributed, scale-out storage service

- 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 Availability Zones (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

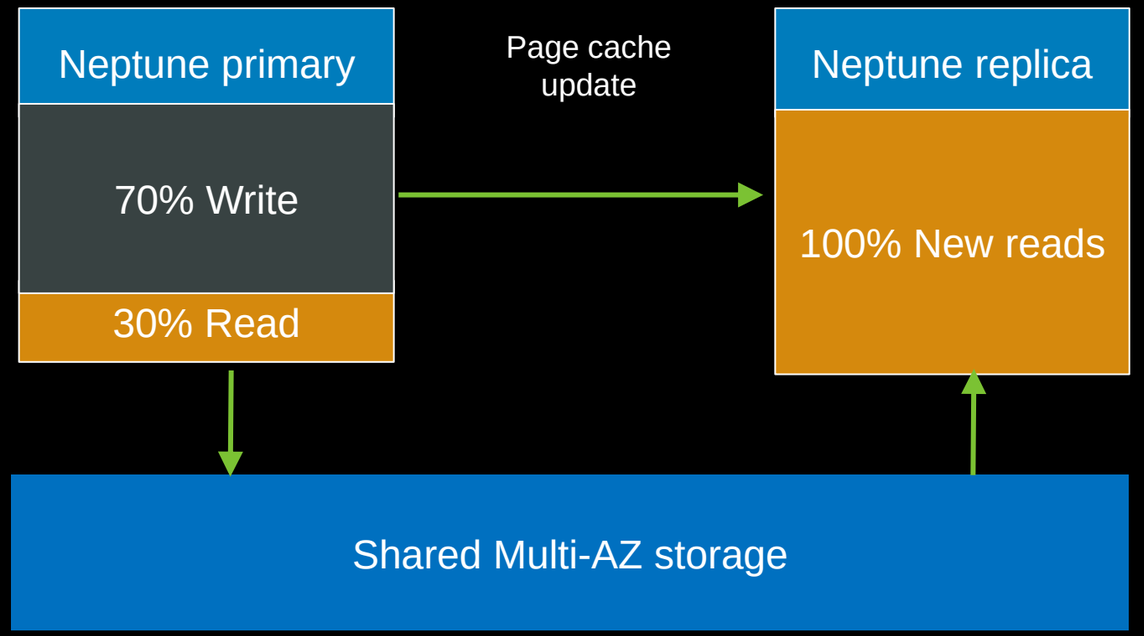
Low replica lag

- 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



Connecting to Neptune with Gremlin or SPARQL

		
Using the web	• HTTP • Web sockets	• HTTP
Using the command line	• Gremlin console • cURL	• RDF4J (OSS) • cURL

Using an application	• Supported language drivers
	  • Official – Java, .NET, NodeJS, Python
	  • Community – Go, Scala, etc.
		• Supported language drivers
		  • Official – Java (Eclipse RDF4J)
		  • Community – Python, etc.

Considerations	• Gremlin Language Variants (GLV) – Allows writing traversals using constructs of the language instead of string concatenation • Need special client for SigV4 signing if IAM authentication is enabled on Neptune • Amazon Neptune Gremlin client driver	• Many third-party open source and commercial tools available for query development • Apache Jena has full support
3rd party tools	https://tinkerpop.apache.org/	https://www.w3.org/2001/sw/wiki/Tools

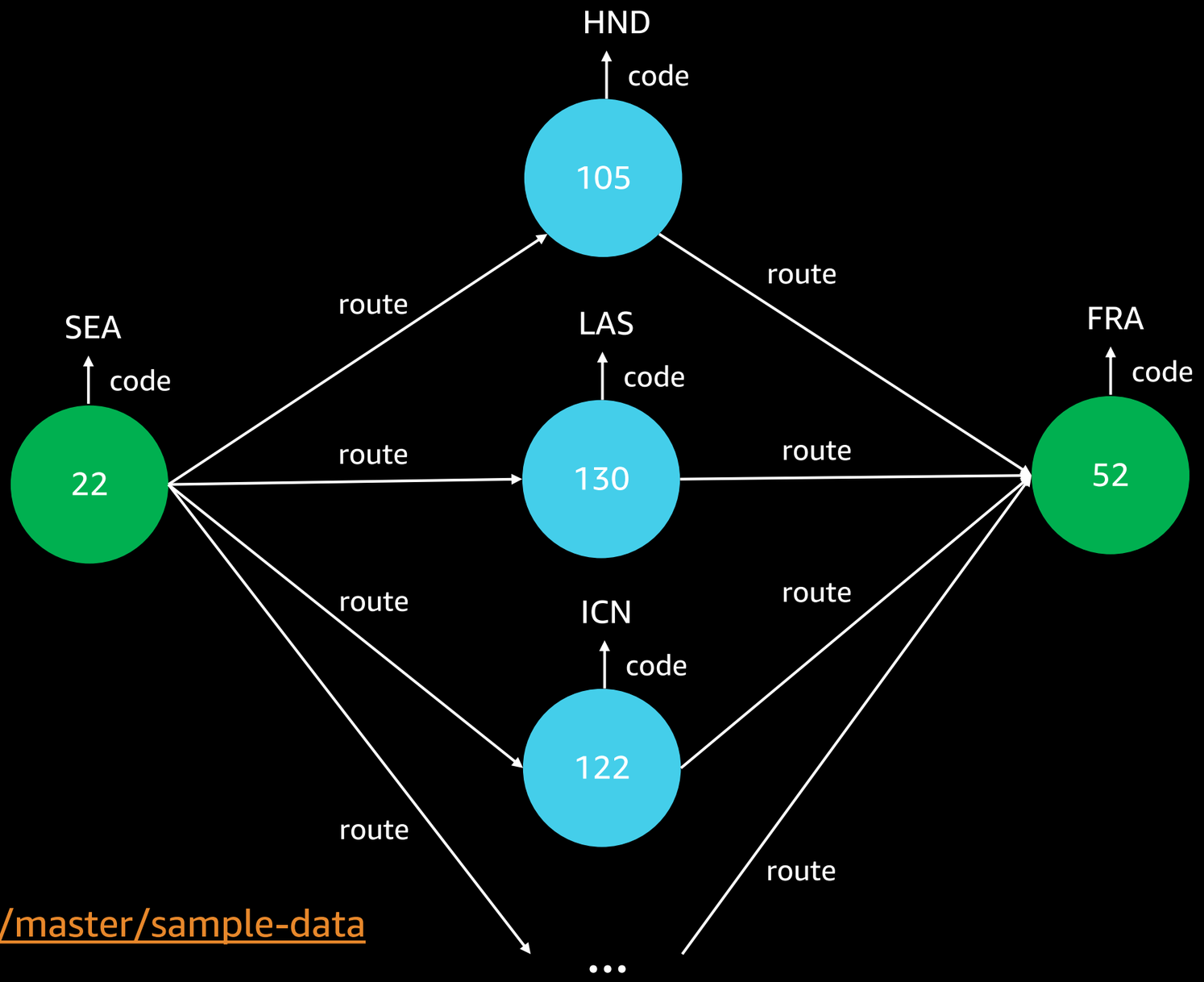
How does Neptune store and query the graph?



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

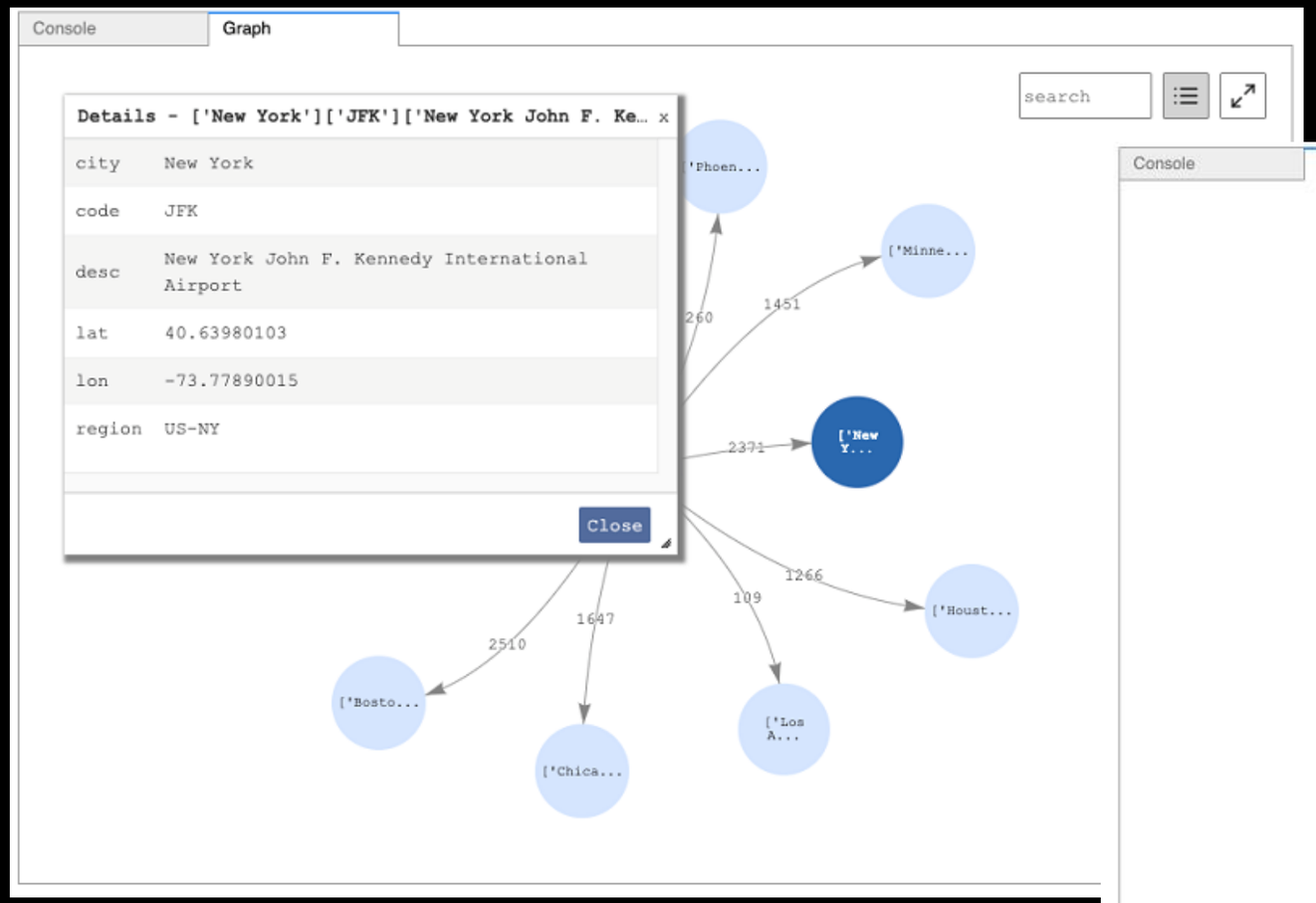
Air routes dataset

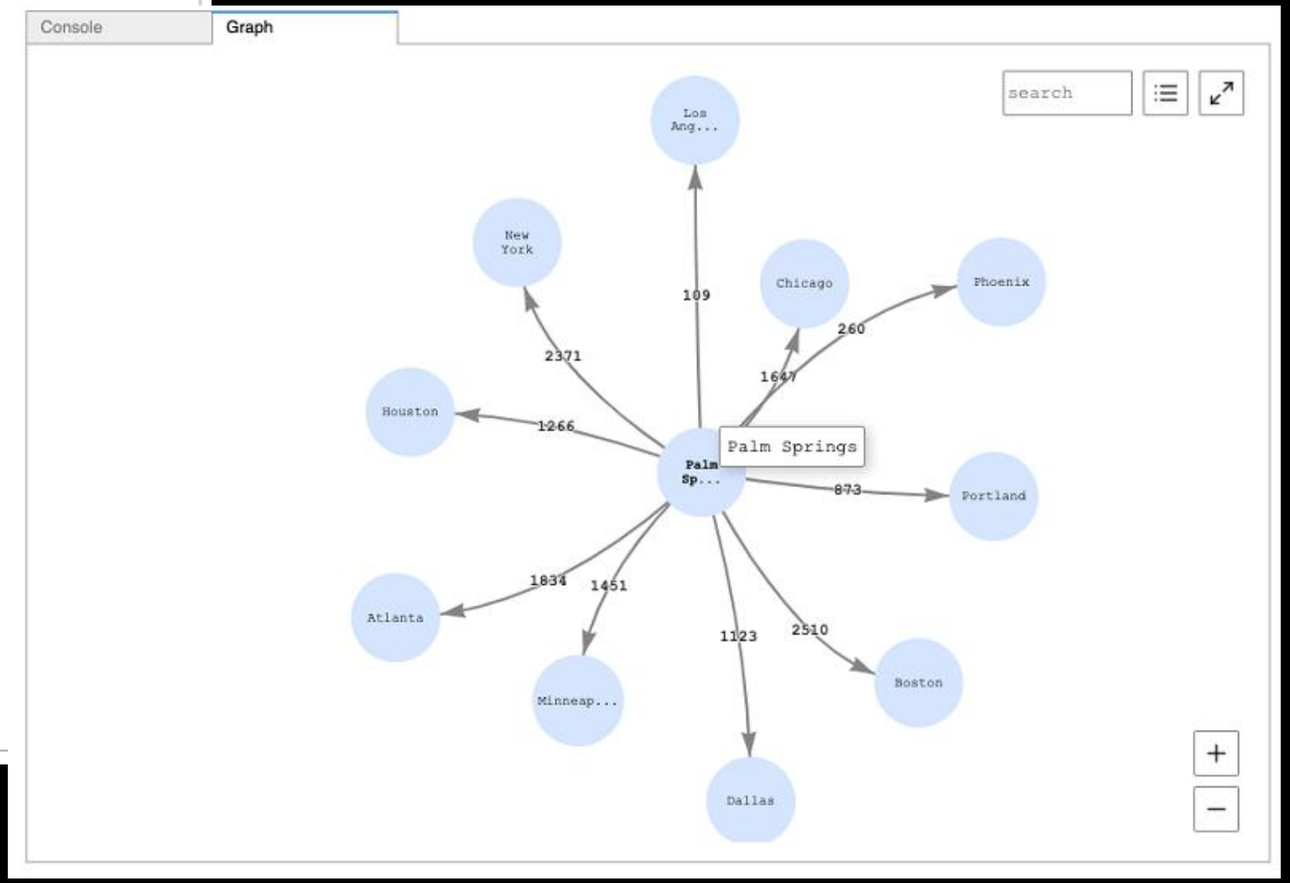
- Models the world's airline route network
- Queries operating over the airport connectivity graph
- Sample queries
  - Given
    - Source and target airport
  - Find
    - All one-stop connections



https://github.com/krlawrence/graph/tree/master/sample-data

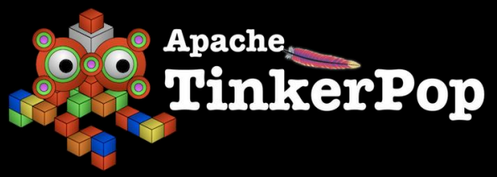
Air routes in the Neptune workbench



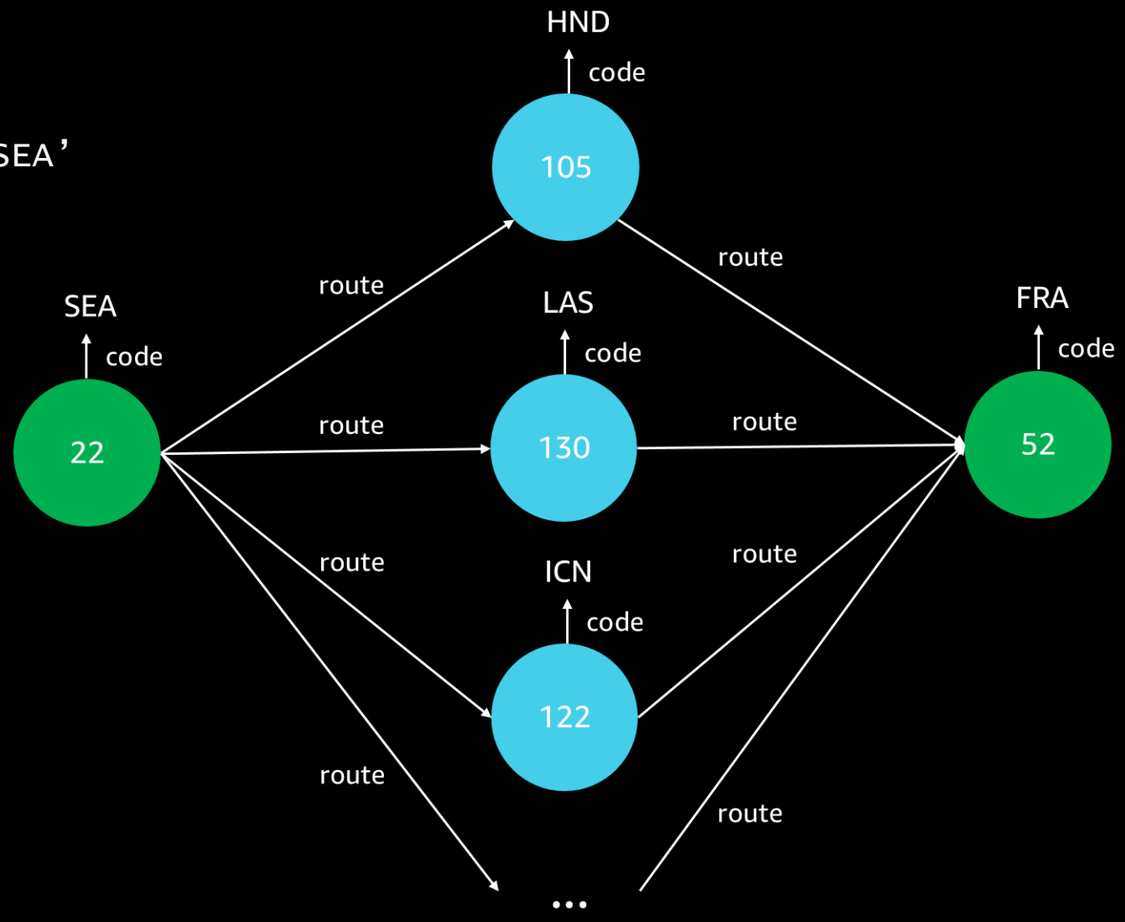


https://aws.amazon.com/blogs/database/visualize-query-results-using-the-amazon-neptune-workbench/

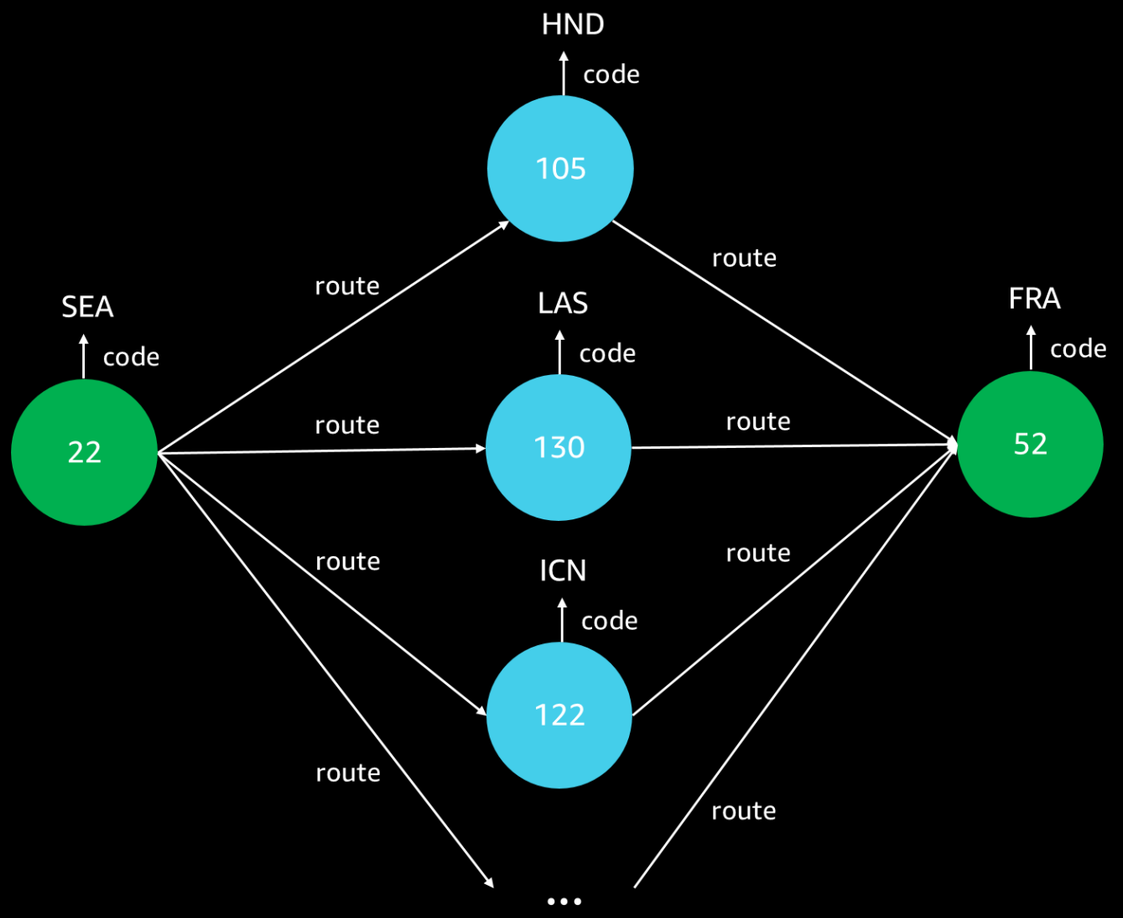
“Find all of the airport codes for one-stop connections from SEA to FRA”




```
# 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

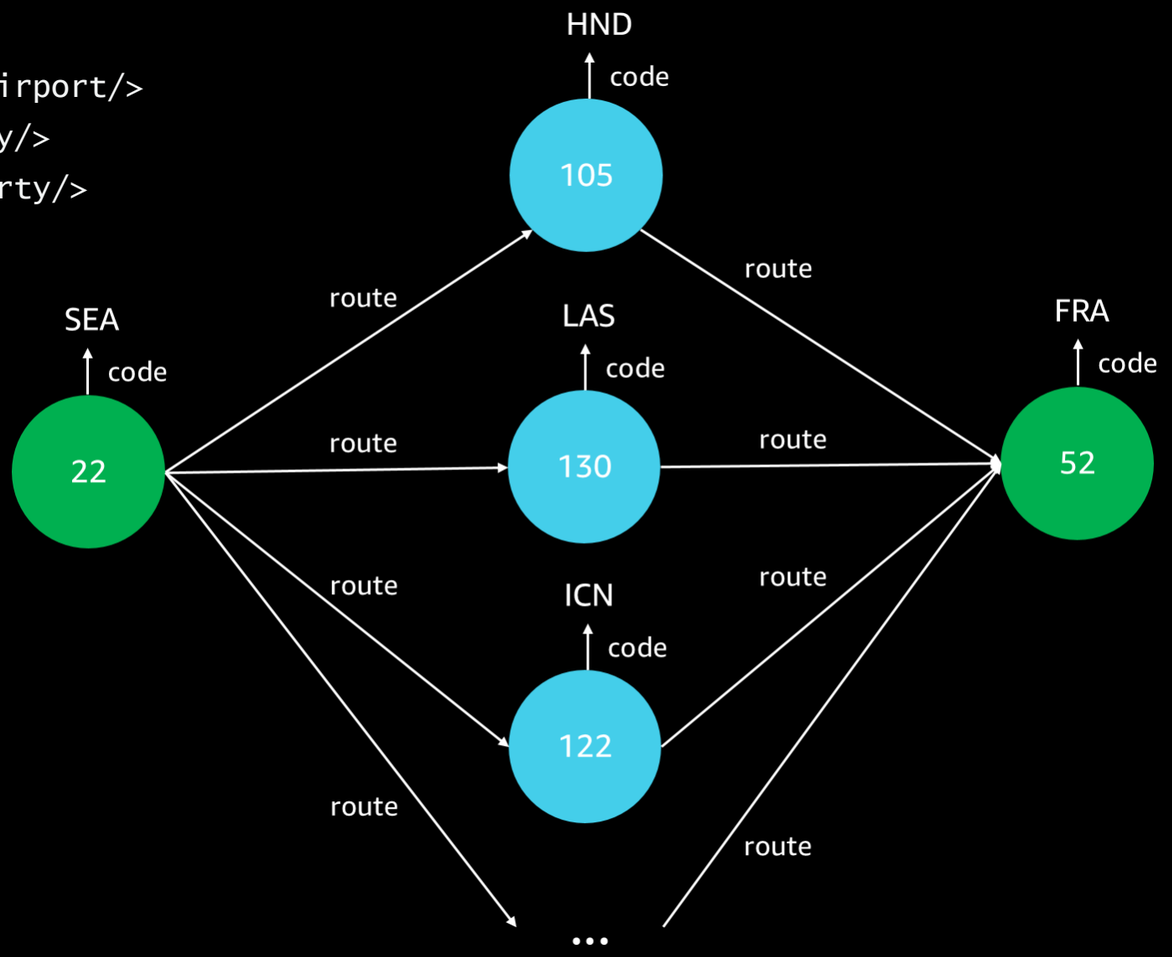


Subject	Predicate	Object	Graph
22	code	"SEA"	default
22	route	105	e1
22	route	130	e2
22	route	122	e3
105	code	"HND"	default
105	route	52	e4
130	code	"LAS"	default
130	route	52	e5
122	code	"ICN"	default
122	route	52	e6
52	code	"FRA"	default

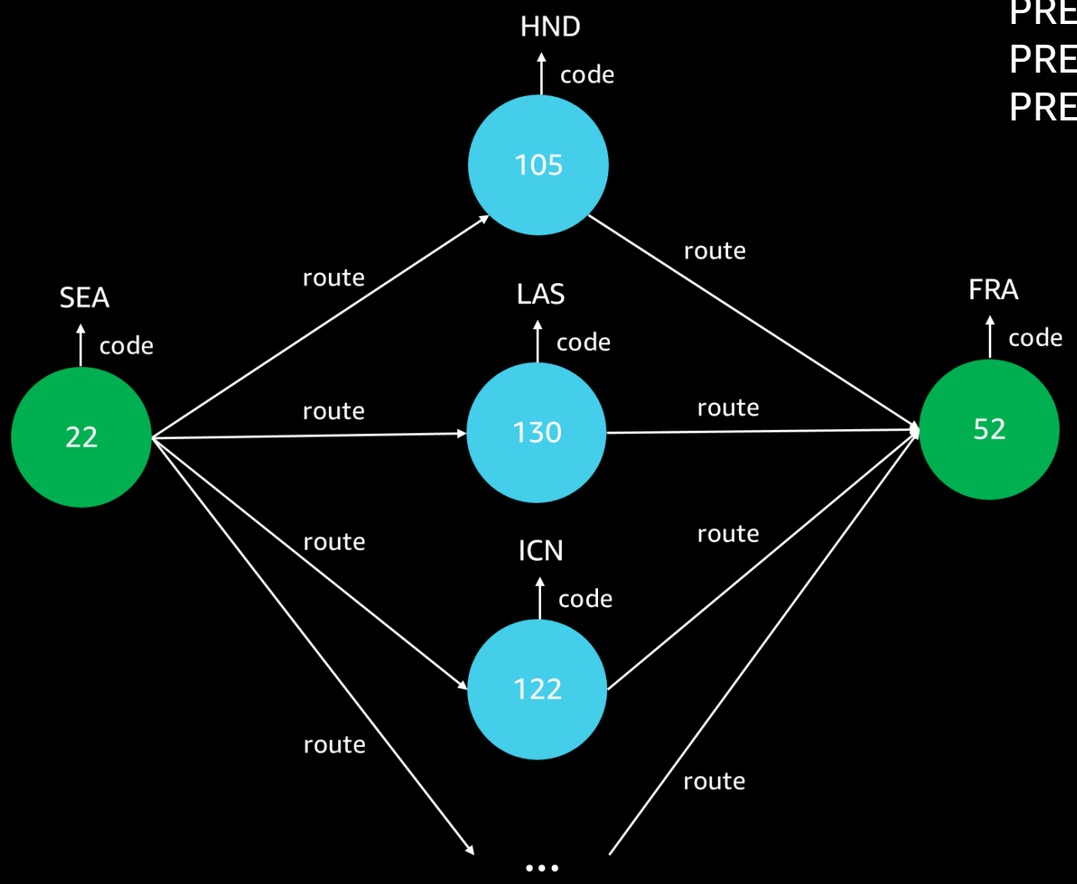
https://docs.aws.amazon.com/neptune/latest/userguide/feature-overview-data-model.html

“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 .
    ?via prop:code ?viaCode .
    ?via edge:route airport:FRA .
}
```



Neptune graph data model




```
PREFIX resource: <http://kelvinlawrence.net/air-routes/resource/>
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/>
```

Subject	Predicate	Object	Graph
airport:SEA	prop:code	"SEA"	default
airport:SEA	edge:route	airport:HND	resource:1
airport:SEA	edge:route	airport:LAS	resource:2
airport:SEA	edge:route	airport:ICN	resource:3
airport:HND	prop:code	"HND"	default
airport:HND	edge:route	airport:FRA	resource:4
airport:LAS	prop:code	"LAS"	default
airport:LAS	edge:route	airport:FRA	resource:5
airport:ICN	prop:code	"ICN"	default
airport:ICN	edge:route	airport:FRA	resource:6
airport:FRA	prop:code	"FRA"	default

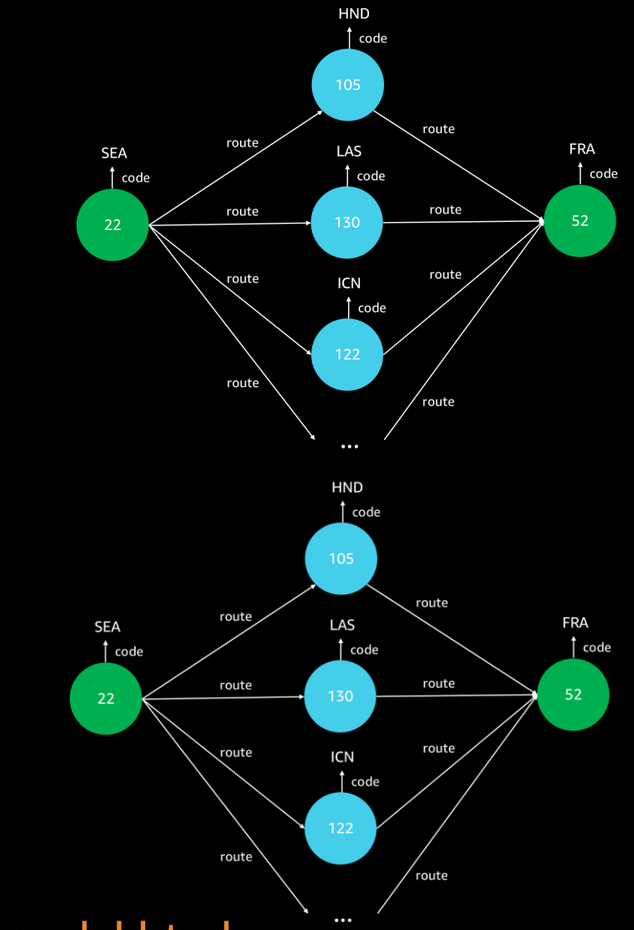
https://docs.aws.amazon.com/neptune/latest/userguide/feature-overview-data-model.html

Neptune graph data model: Indices

Subject	Predicate	Object	Graph

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"*



https://docs.aws.amazon.com/neptune/latest/userguide/feature-overview-data-model.html

Neptune graph data model: Indices

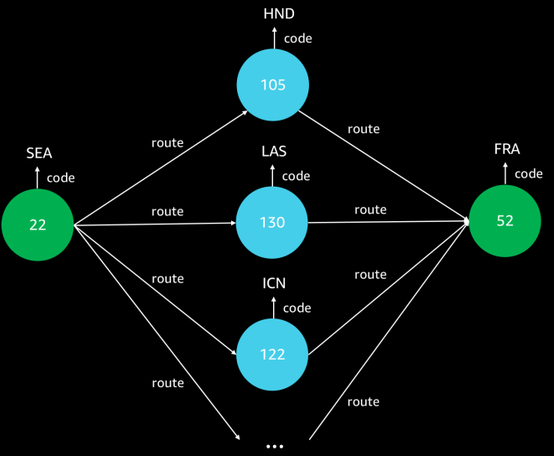
Subject	Predicate	Object	Graph

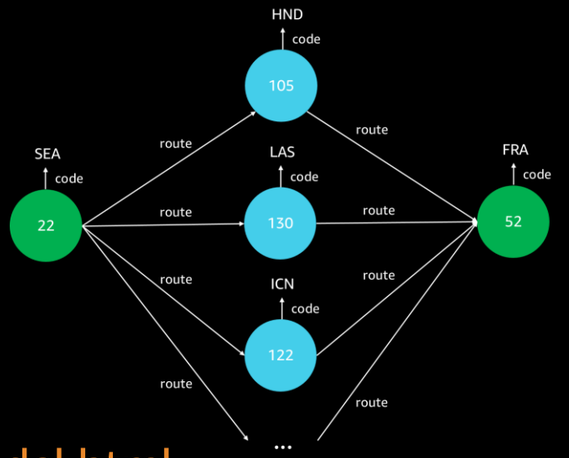
SPOG – Uses a key composed of Subject + Predicate + Object + Graph

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"?*





https://docs.aws.amazon.com/neptune/latest/userguide/feature-overview-data-model.html

Neptune graph data model: Indices



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

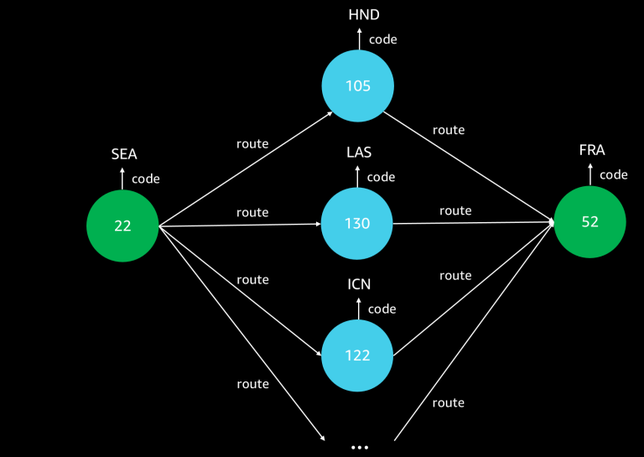
Efficient lookup whenever a prefix of the positions, such as the vertex (subject) or vertex and property identifier, is bound – *Find airport:SEA (22) with code "SEA"*

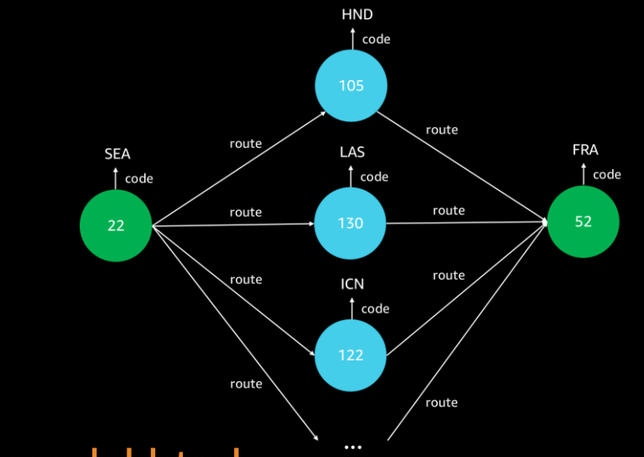
POGS – Uses a key composed of Predicate + Object + Graph + Subject

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 have routes to "FRA"?*





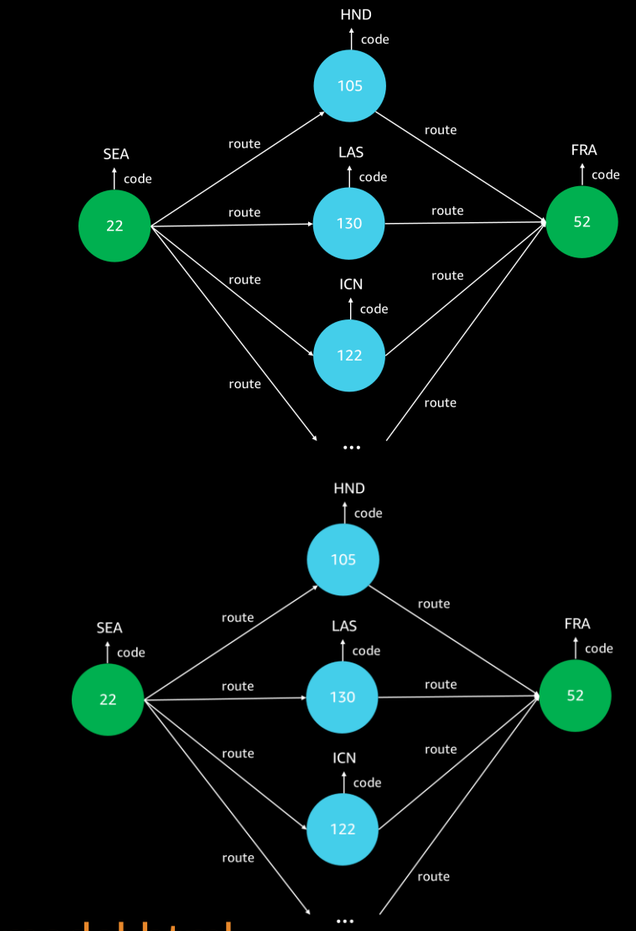
https://docs.aws.amazon.com/neptune/latest/userguide/feature-overview-data-model.html

Neptune graph data model: Indices



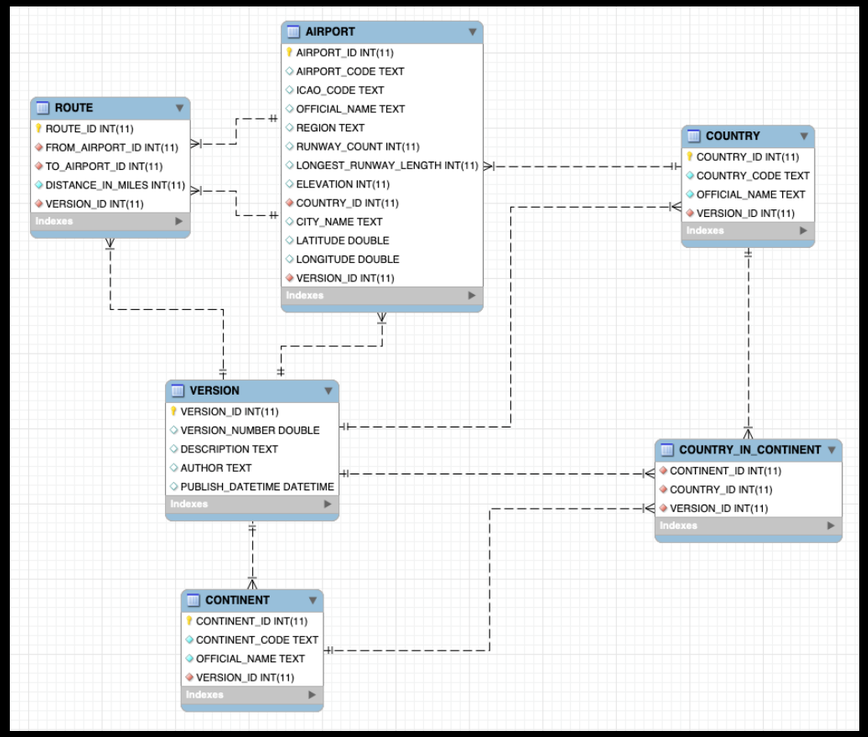
OSGP – Uses a key composed of Object + Subject + Graph + Predicate (Optional – lab mode)

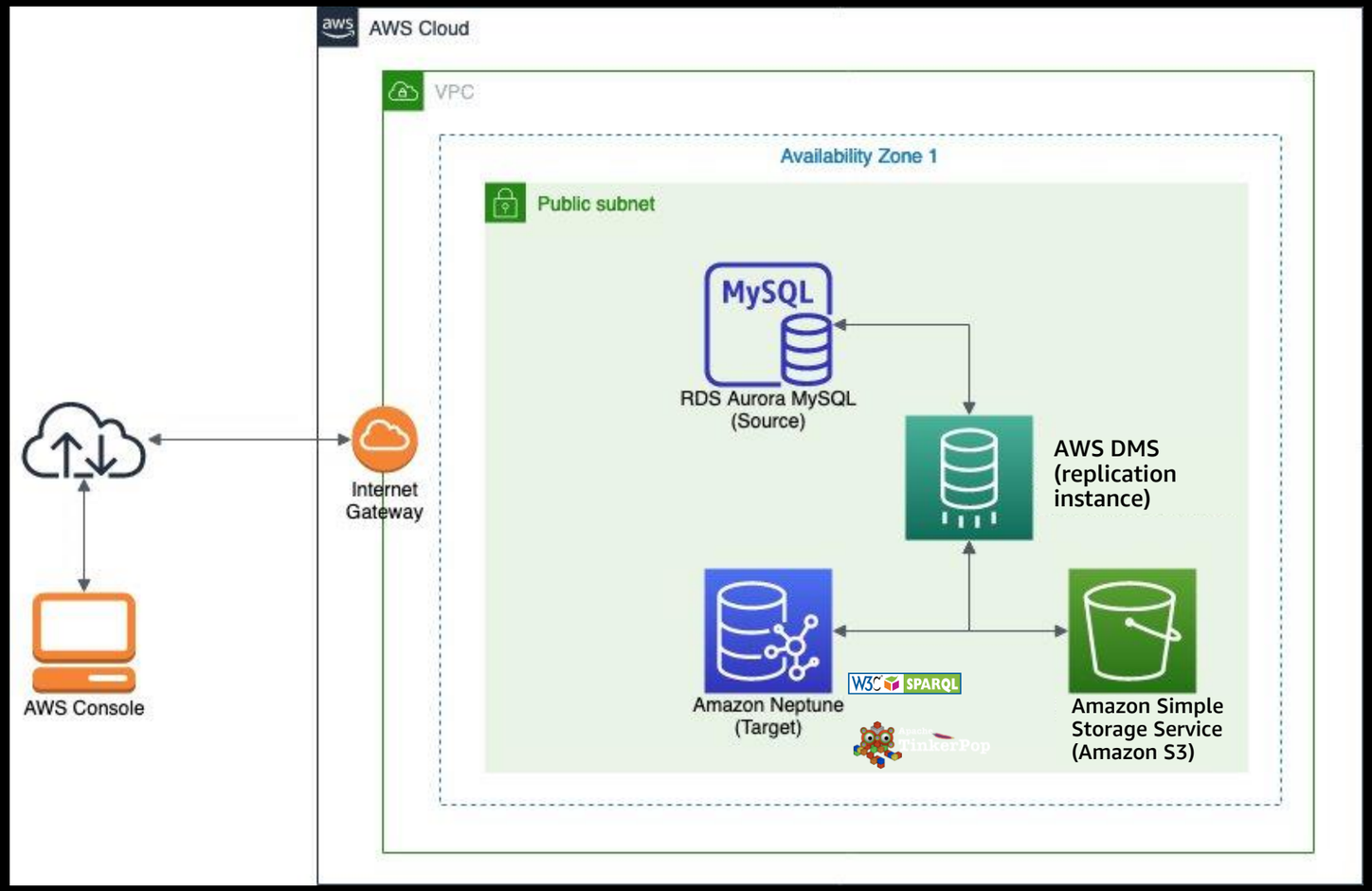
Efficient lookup when the object position is bound – *All nodes or edges with the value of "SEA"*; can be useful if you have a large (10k+) number of predicates



https://docs.aws.amazon.com/neptune/latest/userguide/feature-overview-data-model.html

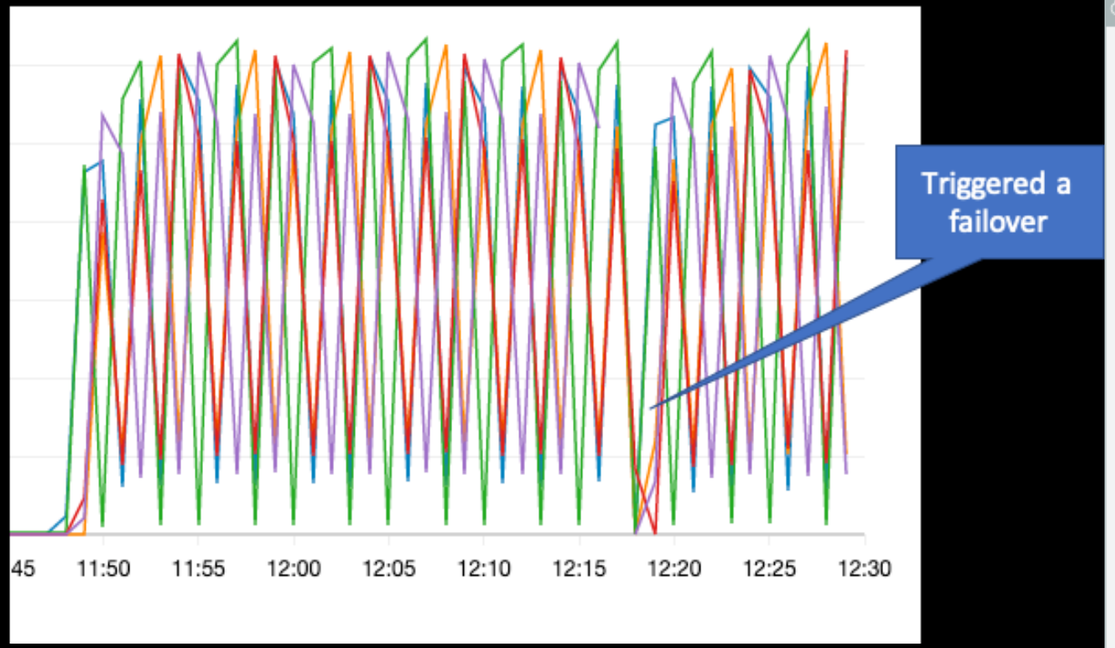
What if the air routes are in your RDBMS?



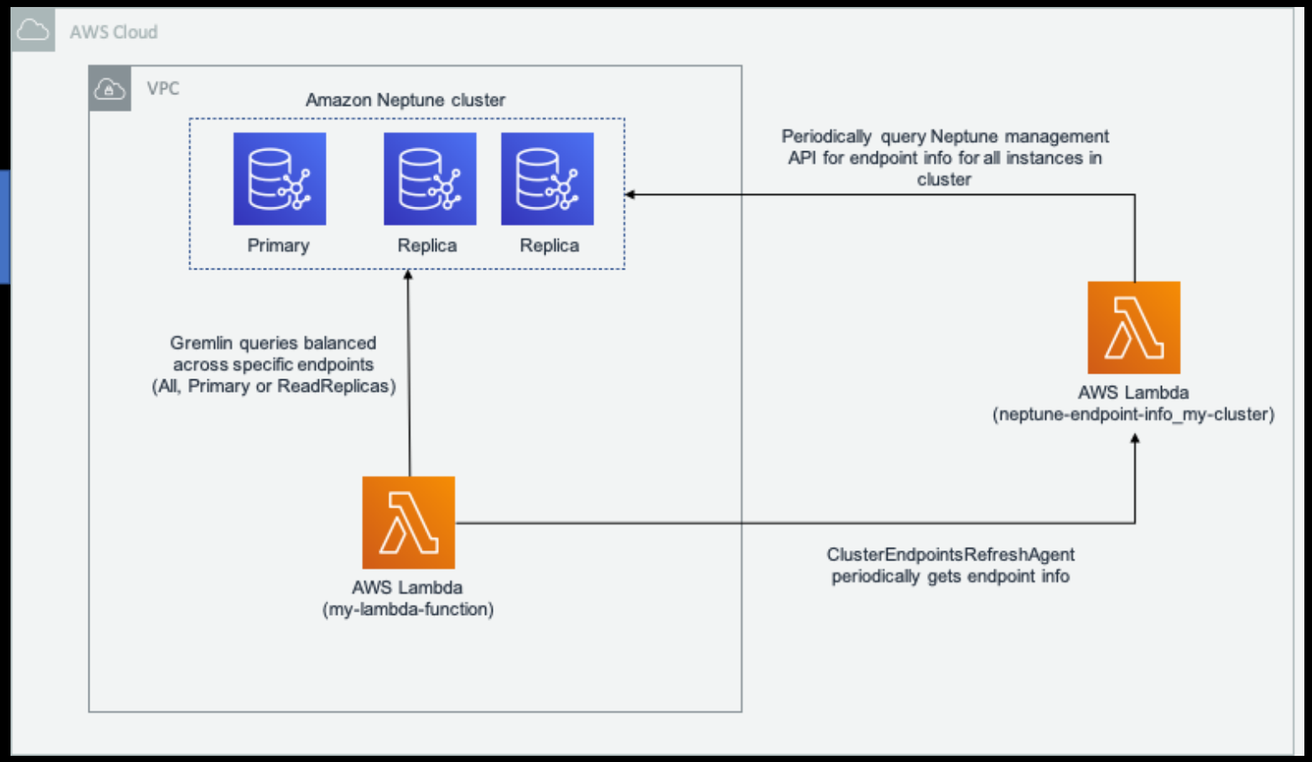


Use the AWS Database Migration Service (AWS DMS) to map and migrate your data

Getting the most out of your Gremlin client



Gremlin requests per second



https://aws.amazon.com/blogs/database/load-balance-graph-queries-using-the-amazon-neptune-gremlin-client/

Related sessions

DAT212

New capabilities to build graph apps quickly with Amazon Neptune

DAT315

ADP's next-generation platform powers dynamic teams with Amazon Neptune

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!

Brad Bebee

b2ebs@

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Please complete the session survey

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.