
AWS
re:Invent
NOV. 27 – DEC. 1, 2023 | LAS VEGAS, NV

DAT325

# Deep dive into Amazon Neptune Analytics & its generative AI capabilities

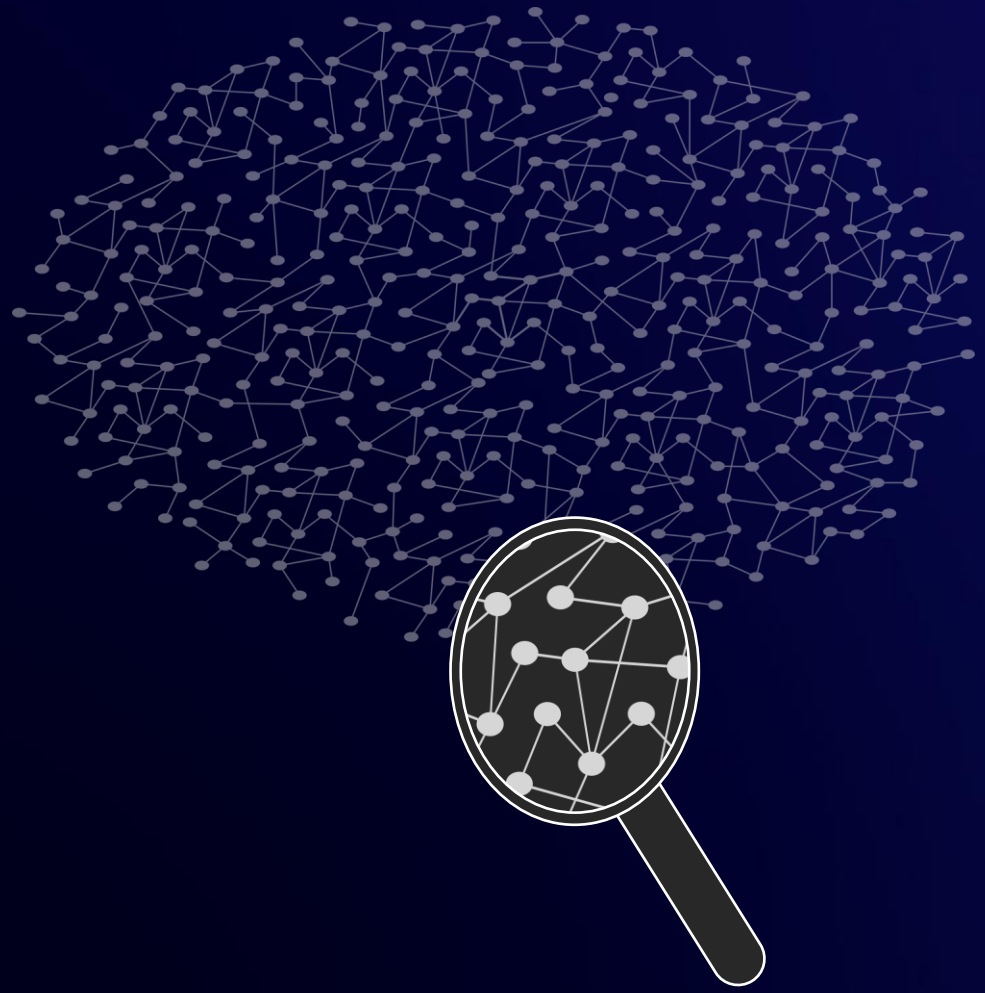
**Brad Bebee**
(he/him)
General Manager,
Amazon Neptune and Timestream
AWS

**Ümit V. Çatalyürek**
(he/him)
Amazon Scholar, Amazon Neptune, AWS
Professor, Georgia Institute of Technology

# Agenda


- Quick review of graphs and use cases
- Understand how Neptune works
- Deep dive into Neptune Analytics
- Show you Neptune Analytics in action
- Take questions!

# Graphs are awesome!



1. Model data based on relationships

2. Applications explore connections and patterns in connected data

3. Processing graphs is hard due to random data access

4. Generalized graph operations require purpose-built processing

# Amazon Neptune

FULLY MANAGED, PURPOSE-BUILT GRAPH DATABASE IN THE CLOUD

Cost-effective

No hardware management

Instant provisioning

Security and compliance

Serverless

Global

- Optimized to **store and map billions of relationships**
- Enables **real-time navigation of connections with millisecond query** response time
- Supports **open standard query languages** openCypher, Gremlin, and SPARQL

# Launched this year

IN CASE YOU MISSED IT!

AWS Graph Explorer

Slow Query Logs

Graph Summary API

Serverless Expansion

Plus 15 engine releases year-to-date to improve performance, features, reliability, and availability…

# Every day thousands of customers use Neptune

**Amazon Neptune**

Customers across different verticals and use cases use Amazon Neptune databases in production today

WIZ · FINRA · SIEMENS · AstraZeneca · JupiterOne · freshworks

amazon alexa · intuit · FACTSET · SAMSUNG · Blackfynn · Rappi

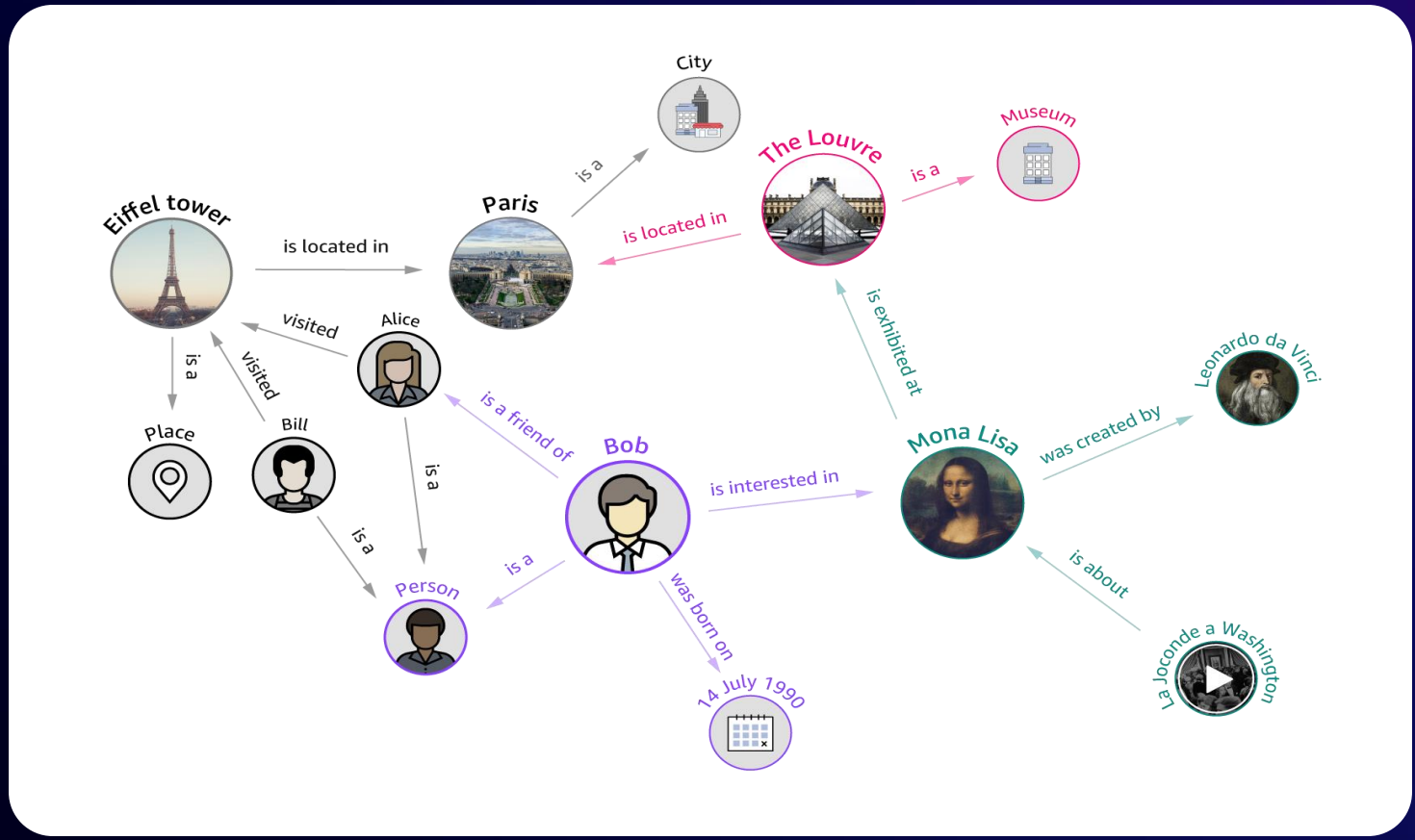
Uber ATG · HUUUGE · NBCUniversal · NETFLIX · asurion · COX AUTOMOTIVE

PaySense · zeta · Pearson · MARINUS ANALYTICS · noonum · THOMSON REUTERS

Case studies: https://aws.amazon.com/solutions/case-studies/?customer-references-cards.q=neptune

# Knowledge graphs

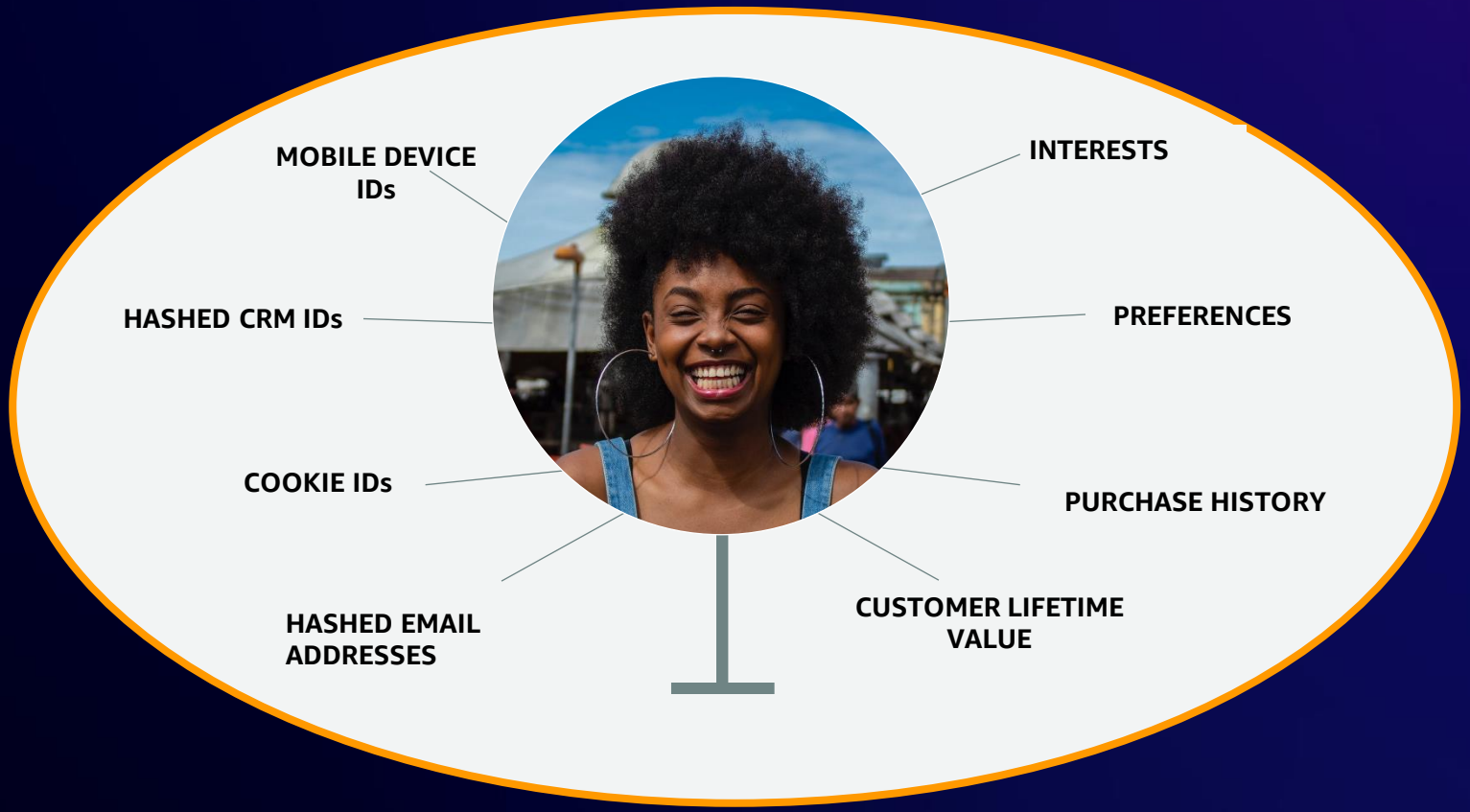
UNDERSTANDING THE WHO, WHAT, WHEN, AND WHERE




https://aws.amazon.com/neptune/knowledge-graphs-on-aws/

# Identity graphs

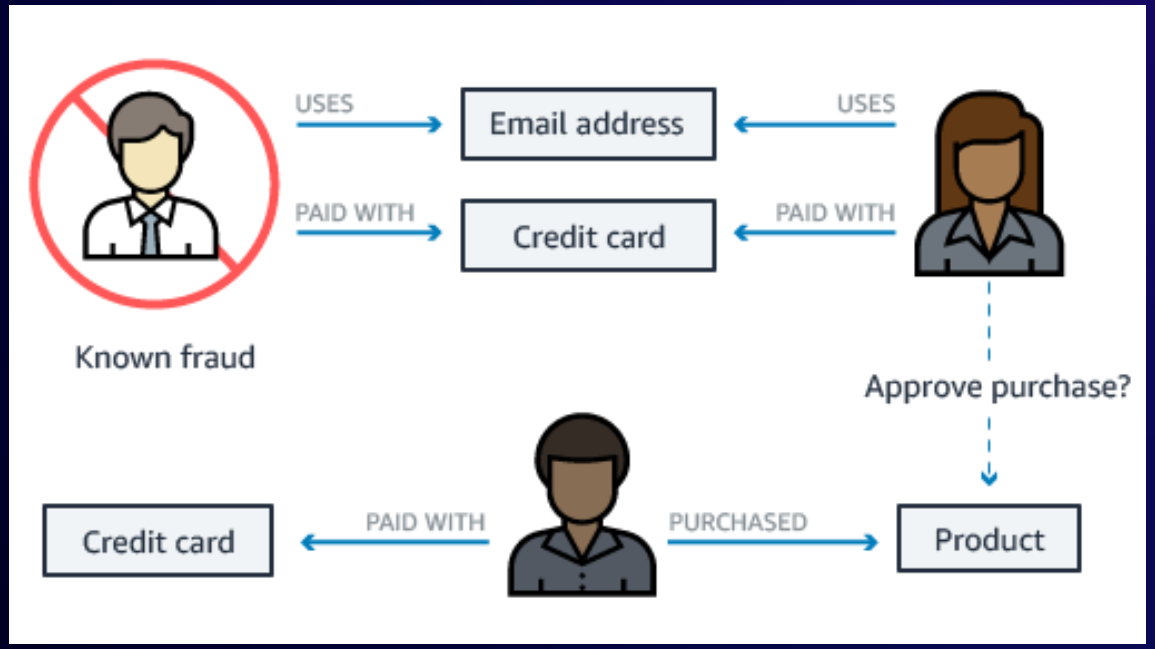
UNIFIED 360° VIEW OF THE CUSTOMER




https://aws.amazon.com/neptune/identity-graphs-on-aws/

# Fraud graphs

DETECTING FRAUD AS IT HAPPENS USING RELATIONSHIPS




https://aws.amazon.com/neptune/fraud-graphs-on-aws/

# Security graphs

UNDERSTAND SECURITY VULNERABILITIES ACROSS LAYERS

1. Cloud Security Posture Management
2. Data Flow/Exfiltration
3. Identity and Access Management

https://aws.amazon.com/neptune/security-graphs-on-aws/

# Wiz security graphs: Detecting critical risks

## Workload context

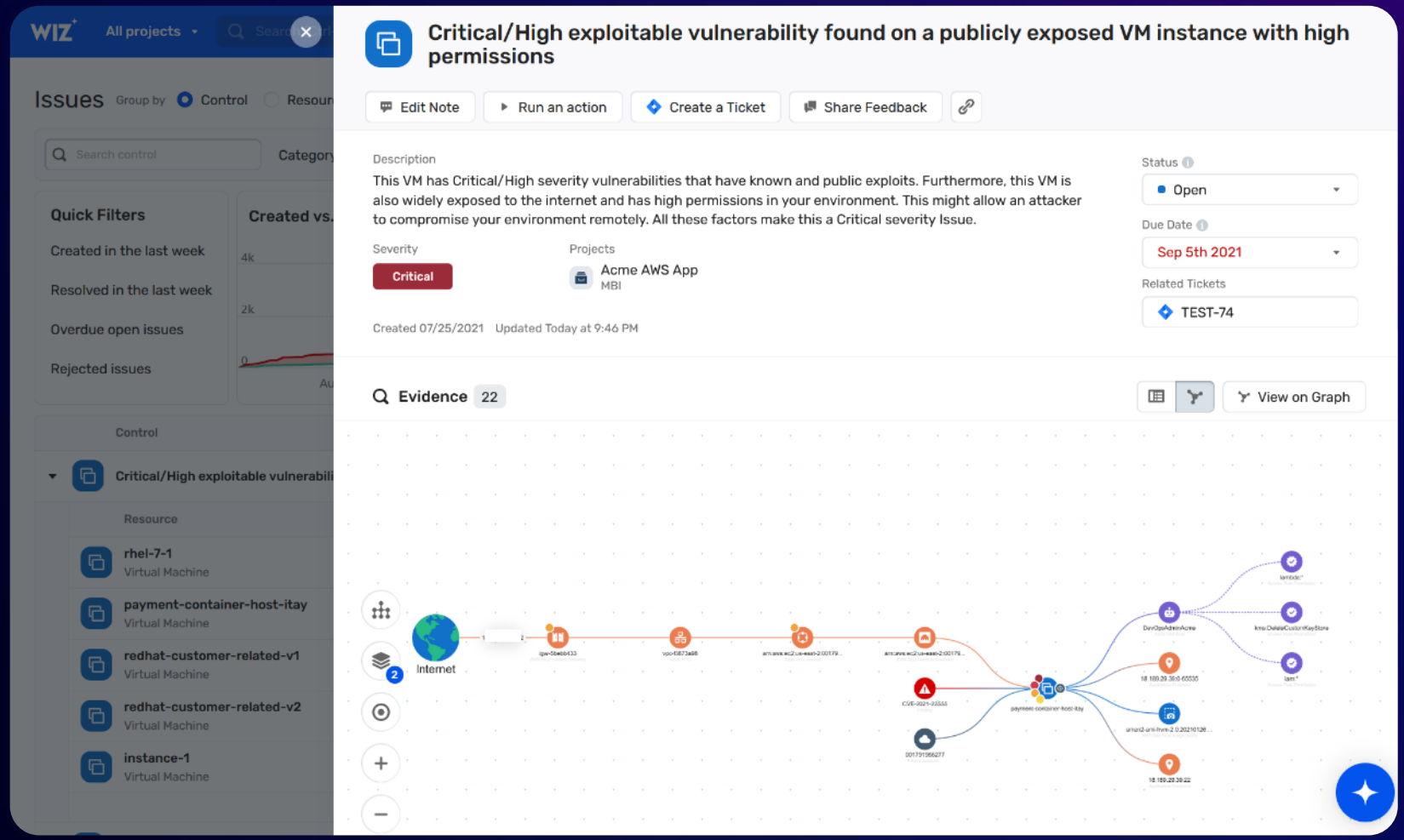
- Vulnerabilities
- Inventory
- Exposed secrets

## Cloud context

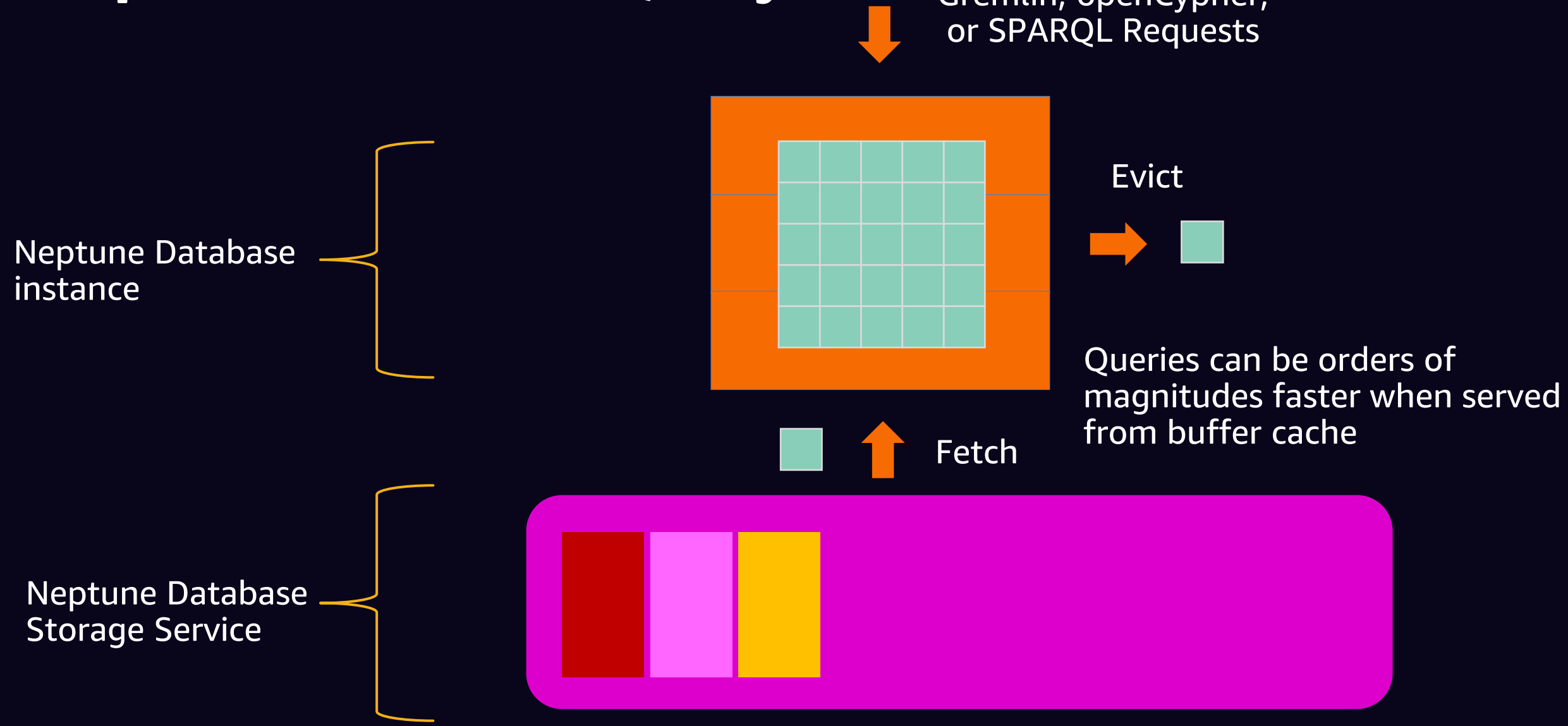
- Resource configuration
- Networking
- Identities

## Business context

- Tags
- Environment
- Business team

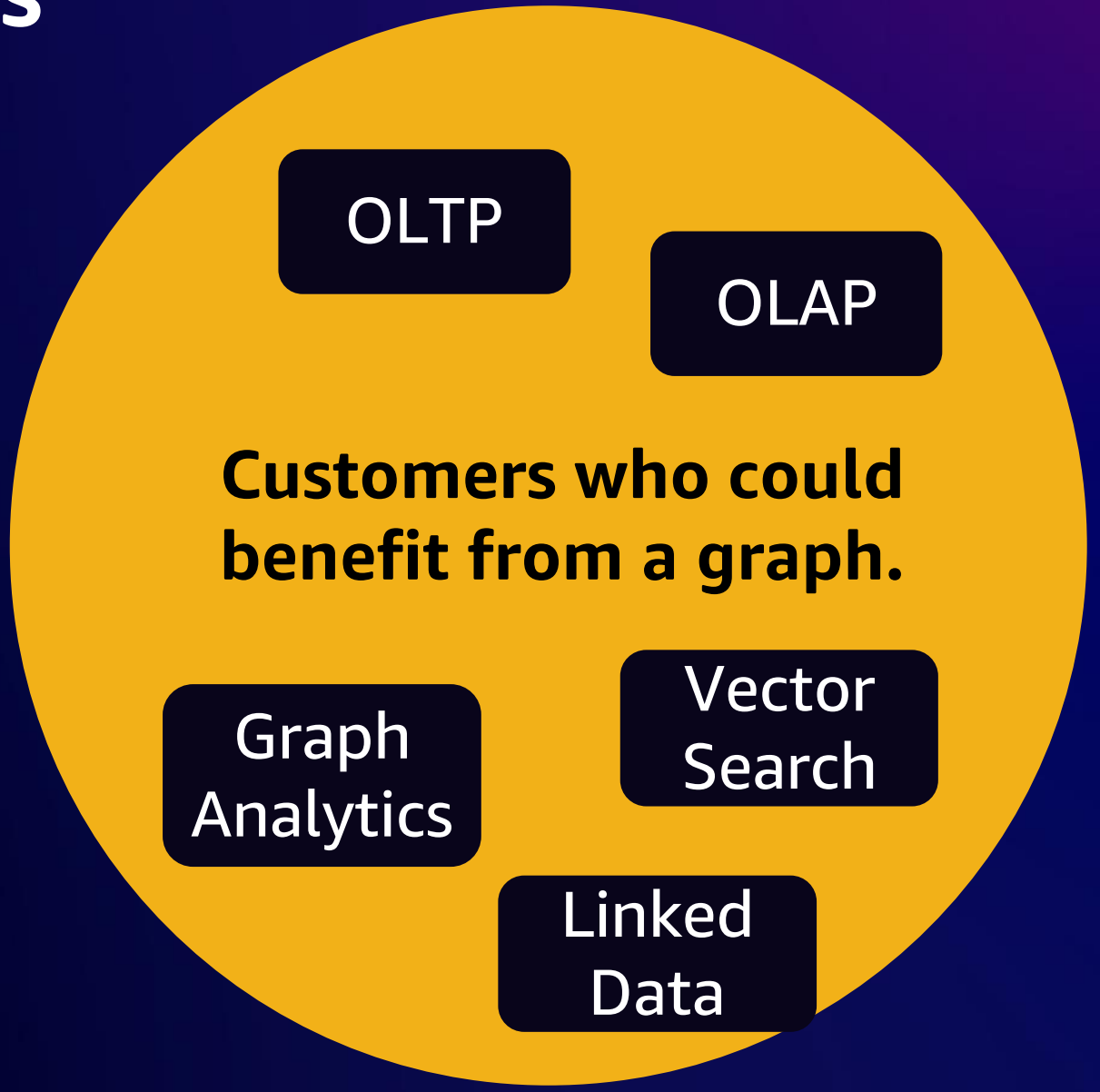


# Neptune Database Query

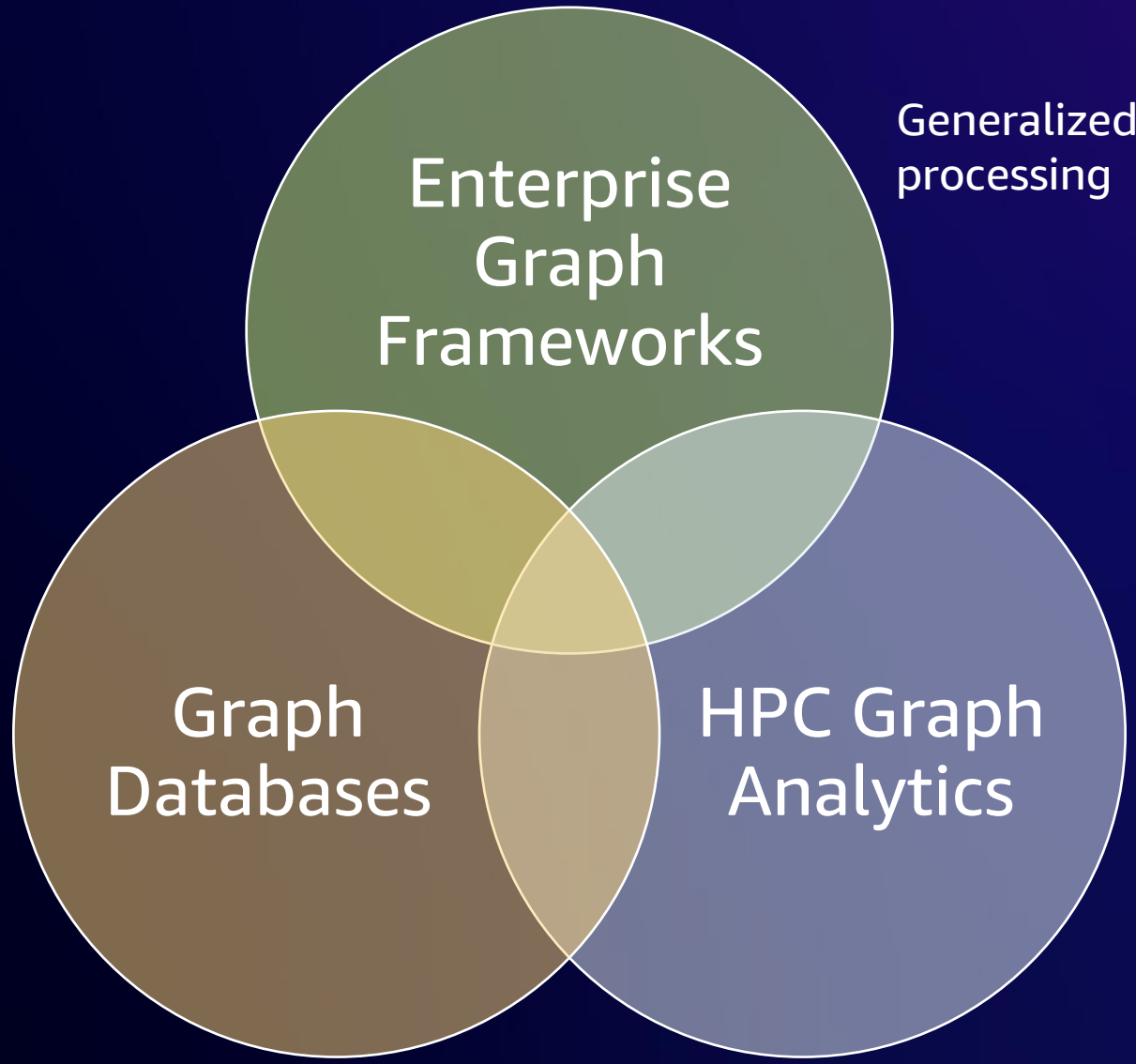


# It’s still Day 1 for graphs


**Customers who know they want a graph database**



# Landscape of current “Graph World”




Generalized APIs for graph processing

Generalized graph query languages with consistency and durability

High performance for specific graph problems

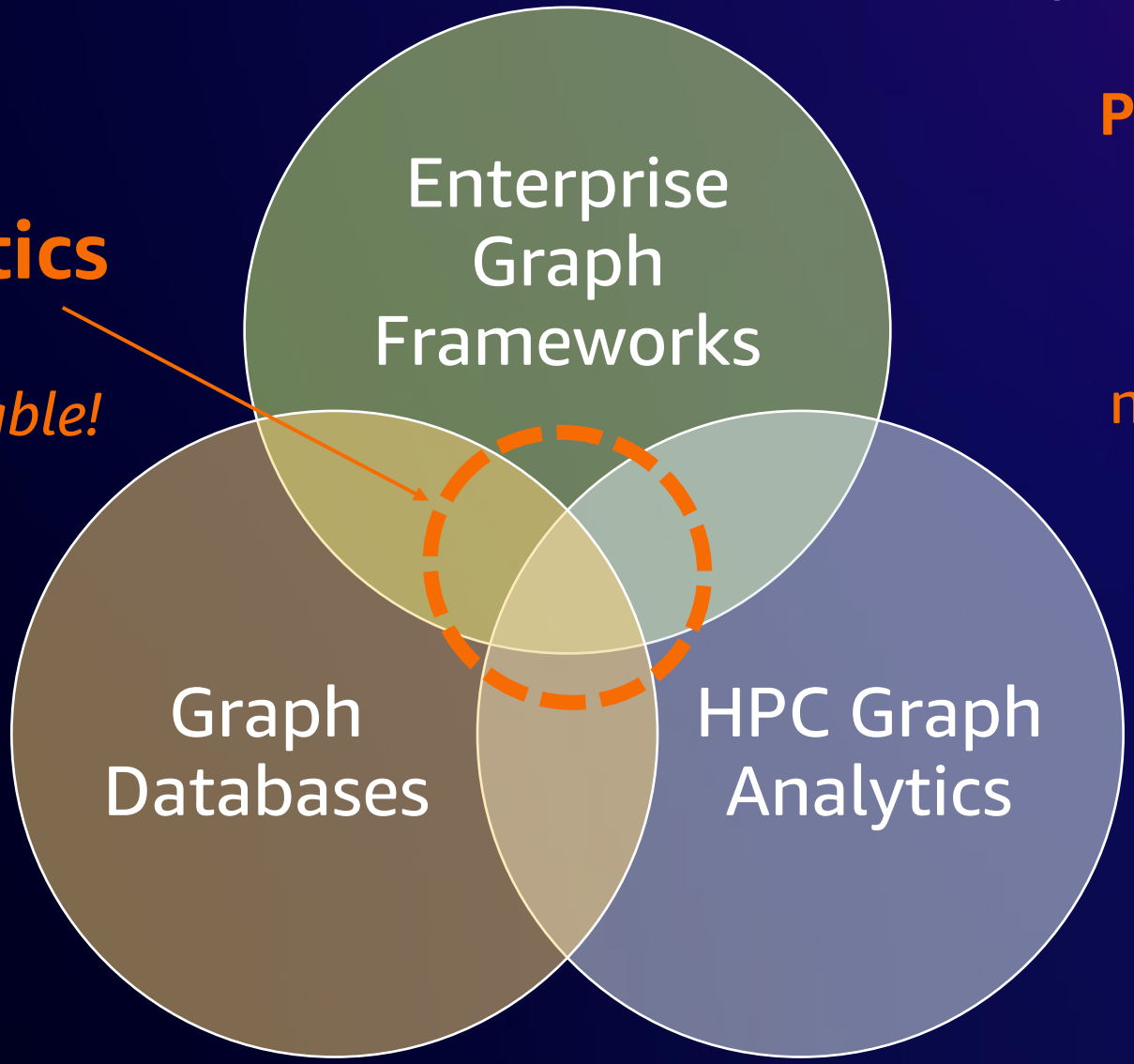
# Customers said they wanted make data discoveries faster by analyzing graph data with tens of billions of connections in seconds

# That means you need to analyze a whole graph in-memory. Quickly!

# Introducing Amazon Neptune Analytics

**Neptune Analytics**

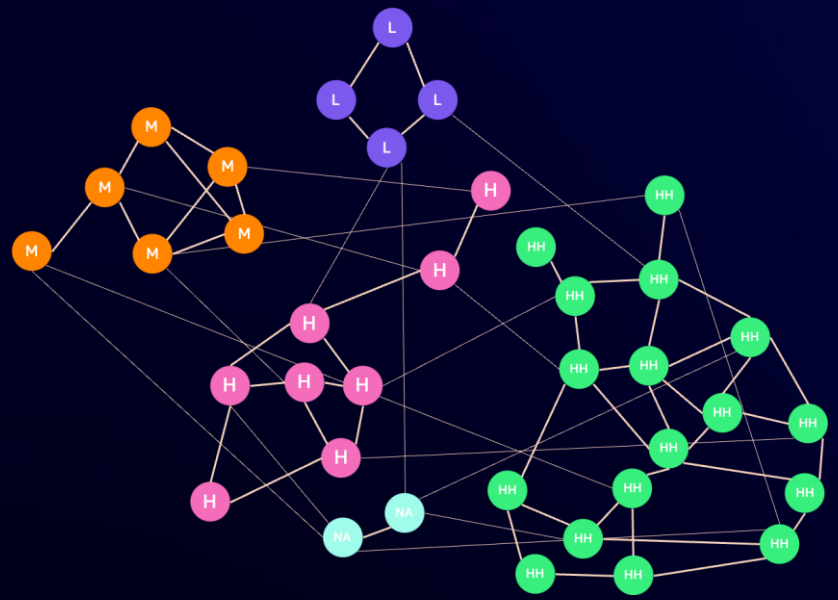
*Now Generally Available!*




**P**erformance is within a **constant factor** of HPC Graph Analytics delivered as a managed service that is easy to use

# Amazon Neptune Analytics

*New analytics engine for Amazon Neptune helps customers to make data discoveries faster by analyzing graph data with tens of billions of connections in seconds*




Example of Clustering Analysis

## Single Service for Graph Workloads

- Invoke popular graph algorithms, low-latency queries, and vector searches with a simple API using openCypher, a popular open-source graph query language
- Create graphs from a Neptune Database or by loading data from Amazon S3

## High-performance graph analytic queries and graph algorithms

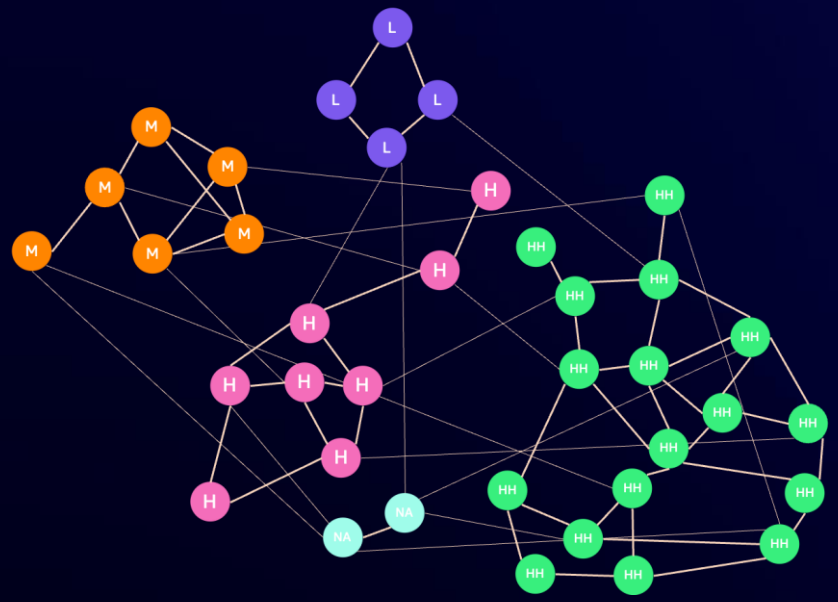
- In-memory with low-latency responses, high throughput, and fast loads
- HPC-style graph partitioning
- Log-based durability model with transaction support

## Store and Search Vectors for Generative AI Applications

- Storing and search embeddings trained from an LLM in graph queries
- Natural language graph queries via Neptune's LangChain integration

# Amazon Neptune Analytics

*Use cases for fast analysis of large graphs, high-throughput analytical queries, and vector search*




Example of Clustering Analysis

## Ephemeral Analytics

Customers load and analyze large datasets quickly

## Low Latency Analytical Queries

Extract insights from graph data and using novel features for ML models to serve low-latency graph analytics to increase user engagement

## Vector Search with Graph Data

Build generative AI applications using techniques like retrieval augmented generation (RAG) that use graphs and vector search for context augmentation

# Algorithms: Callable from openCypher Queries

## Clustering

- Weakly Connected Components
- Label Propagation
- Strongly Connected Components

## Similarity

- Common Neighbors
- Total Neighbors
- Overlap Similarity
- Jaccard Similarity

## Centrality

- Degree
- PageRank
- Closeness

## Path Finding

- Breadth First Search
- Single Source Shortest Path
- topK Hop-Limited BFS

## Vector Similarity

- topK Search
- Vector Distance

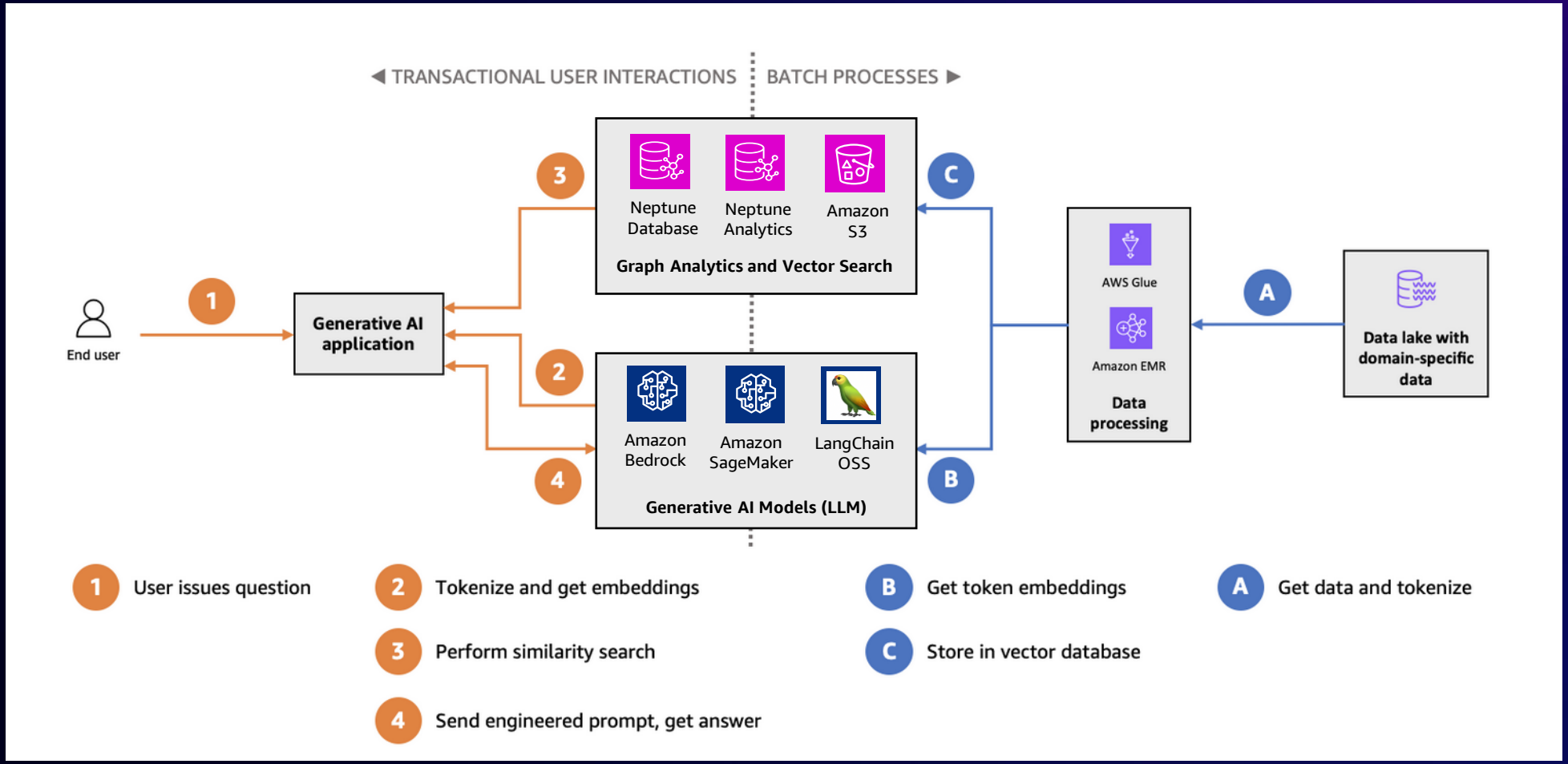
# Using Algorithms in Neptune Analytics

```
// Algorithms
MATCH (n:airport {country: 'US’})

WITH collect(n) as airports, n.region as region

CALL neptune.algos.bfs.levels(n)

YIELD node, level

RETURN node, level
```

# Neptune Analytics: Vector Search + Graph Query

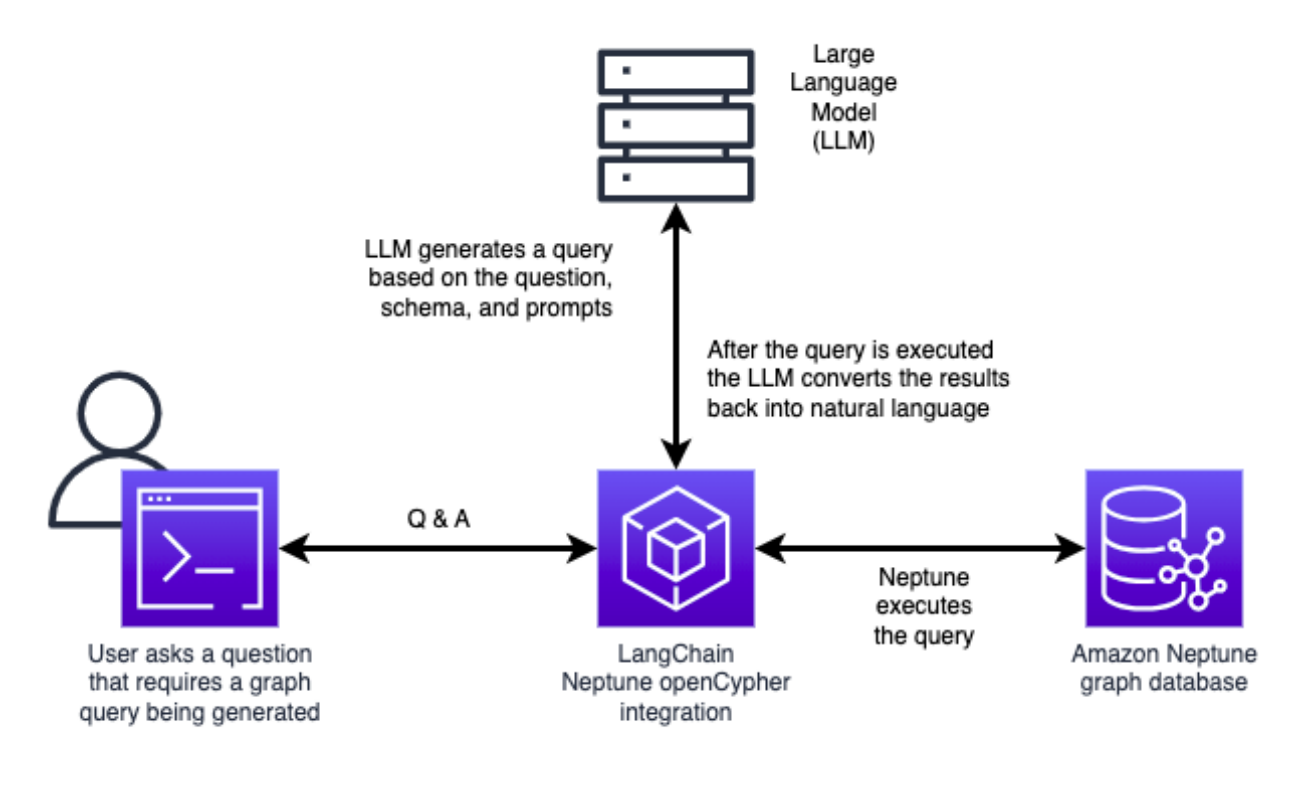
```
MATCH (n:Book {name: 'Travel: Portugal'})

// 1 //
CALL neptune.vectors.topKByNode(n, { topK: 10 } )
YIELD node, score, rank

// 2 //
MATCH p=(node)-[*1..3]->(suspicious)

WHERE (suspicious: seller OR suspicious: lister OR suspicious: buyer)

// 3 //
RETURN n, collect(p), score, rank

ORDER BY rank DESC
```

# Deploying Neptune Analytics in a generative AI application



# Amazon Neptune & LangChain integration

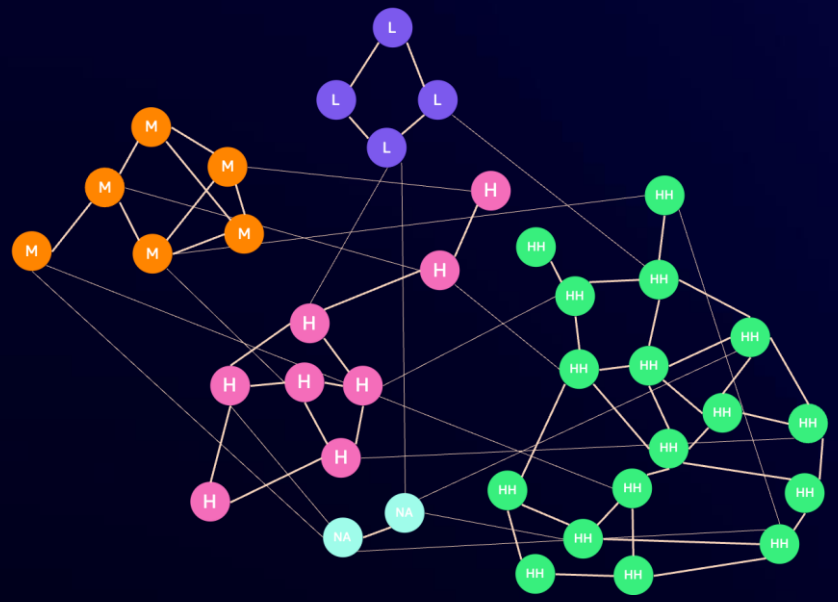
*Query a Neptune graph using the English Language and return a human readable response*




```
from langchain.chat_models import Bedrock
from langchain.chains import NeptuneOpenCypherQAChain
from langchain.graphs import NeptuneGraph

graph = NeptuneGraph(host= "<neptune-host>",
                     port=8182,
                     use_https=True)

llm = Bedrock(model_id = “anthropic.claude-v2”)

chain = NeptuneOpenCypherQAChain.from_llm(llm=llm,
          graph=graph, verbose=True, top_K=10,
          return_intermediate_steps=True,
          return_direct=False)

chain.run("how many outgoing routes \
          does the Austin airport have?")

> 'The Austin airport has 98 outgoing routes.'
```

# Amazon Neptune Analytics

*Ephemeral Analytics Customer Examples*



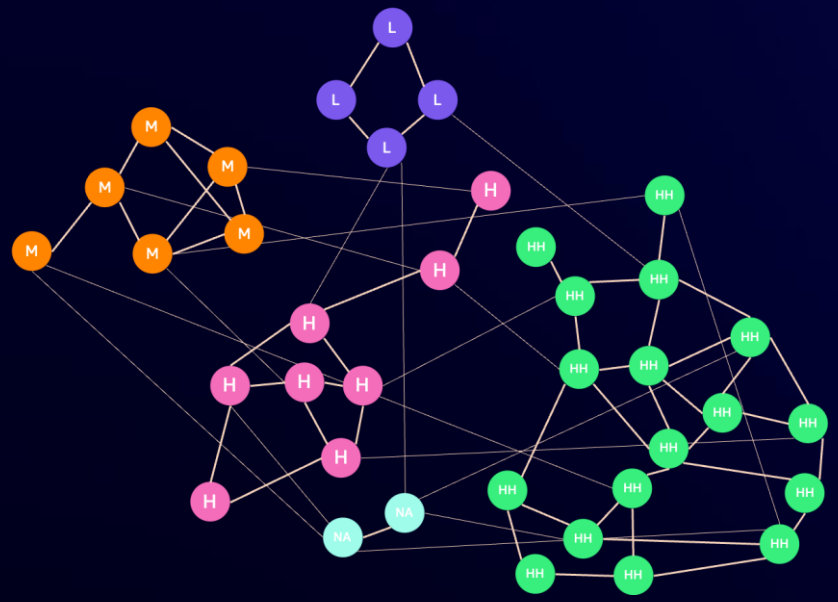

Example of Clustering Analysis

A financial services company provides real-time fraud detection at point of sale; they use ephemeral graph analytics to identify other leading indicators of fraud to increase point of sale intervention from 17% to 58%

A large media and technology company loads large graphs quickly, performs analysis, and turns off the service for TCO reduction and deployment simplification of data science pipelines

# Amazon Neptune Analytics

*Low Latency Analytical Queries*



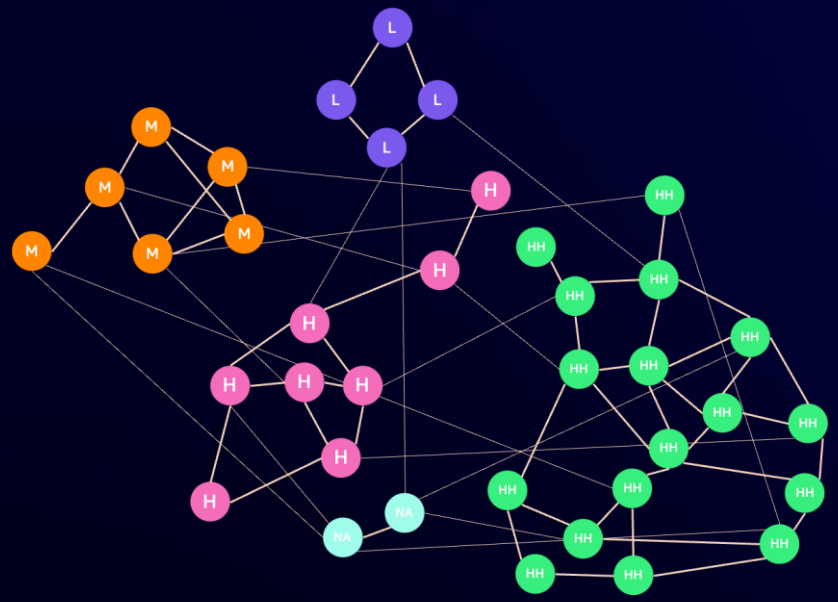

Example of Clustering Analysis

Snap, an instant messaging app with more than 750 million monthly active users, is using Neptune Analytics to perform graph analytics on billions of connections in seconds to enable friend recommendations in near real-time

Amazon.com reduced time to resolution by 25% on fraud investigation cases by augmenting their cases with connections from a graph

# Amazon Neptune Analytics

*Vector Search with Graph Data*



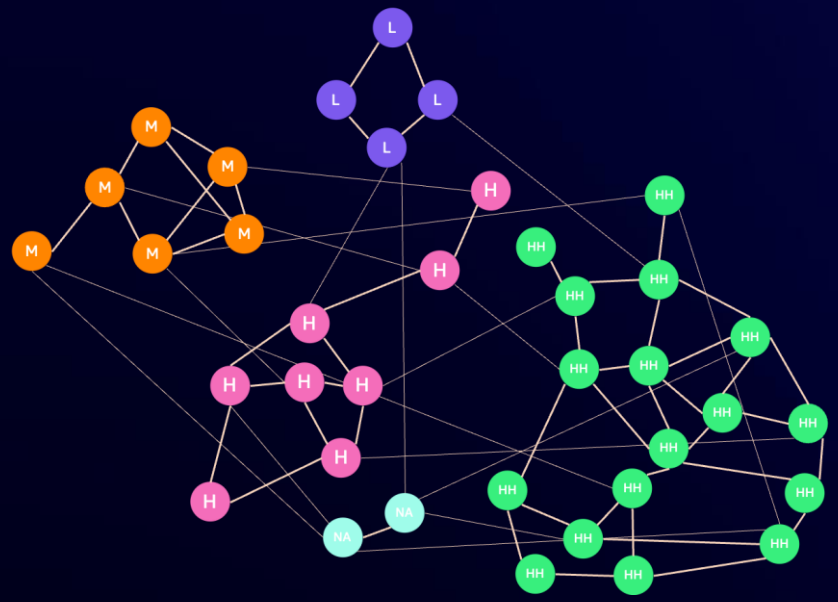

Example of Clustering Analysis

A large healthcare products company is building scientifically aware search by augmenting their proprietary knowledge graph with vector similarity search to discover new products

An online retail store needs to use known pirated material to quickly identify similar media in conjunction with a knowledge graph to identify patterns of deceptive listing behaviors to find malicious sellers

# Amazon Neptune Analytics

*Customers want to simply manage graphs and pay for what they use*




Example of Clustering Analysis

## Pricing Model

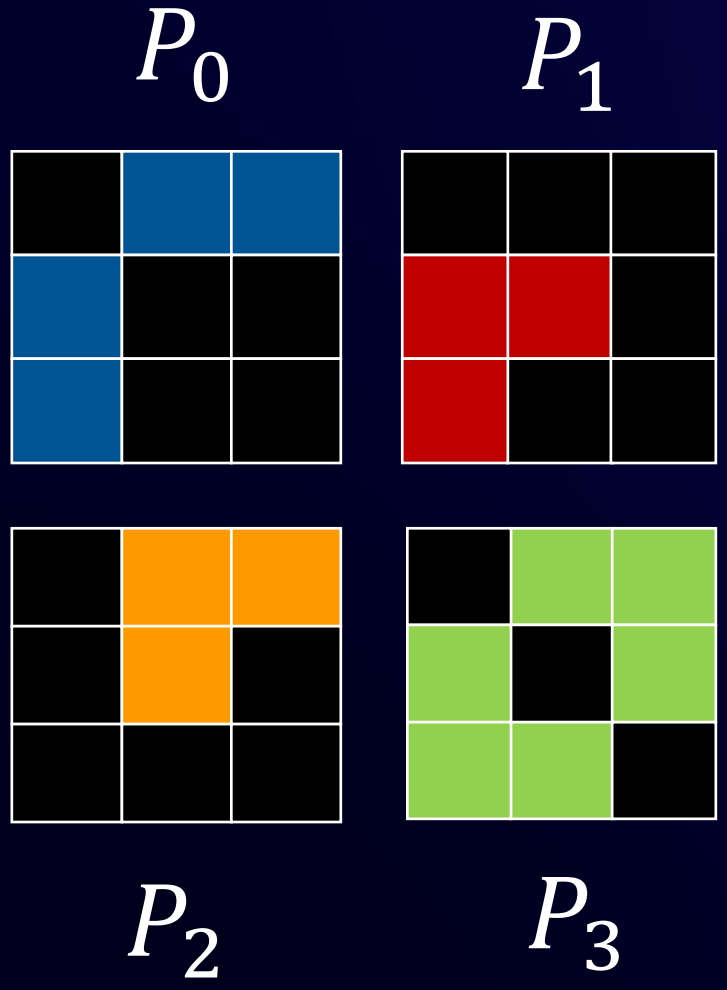
- Memory-optimized Neptune Capacity Units (m-NCU)
- m-NCUs provide a per hour price for the capacity of memory provisioned with associated compute and network resources
- $ per m-NCU-hour price

## Customer Experience

- Simplifies graph creation: customers don’t think about instances
- Faster time to “graph analysis”!

# Peek inside Neptune Analytics

# Neptune Analytics Graph Partitioning



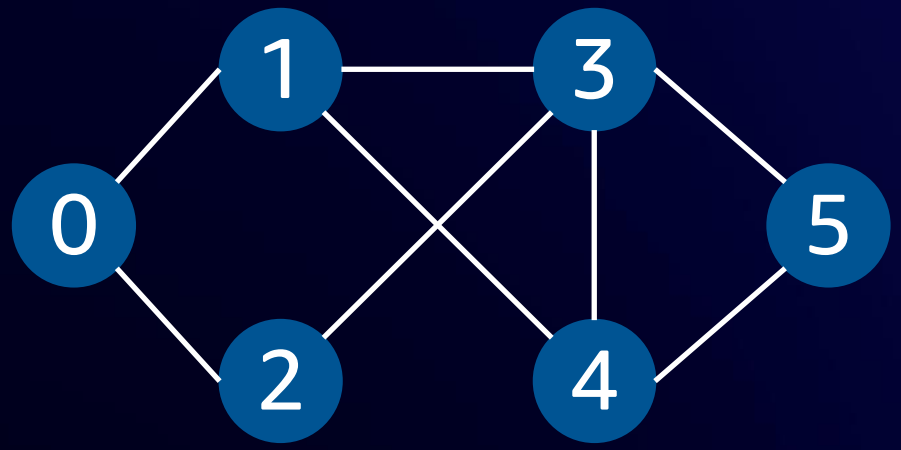

- Partitioning and memory-optimized architecture provides superior load and scan rates with transactional support
  - 80X faster loads and scans (10M/sec)
  - 200X faster columnar scans (100M/cols/sec)
- Three different kinds of partitioned relations:
  - Dictionary partitions, encoding lexical forms to global identifiers
  - Topology partitions, which represent the structure of the graph
  - Property value partitions of the node/edge property sets
- Relations are partitioned according to a mixture of techniques
  - Hash partitioned dictionary, 1D partitioned vertex properties, 2D partitioned edges with co-located edge properties, etc.

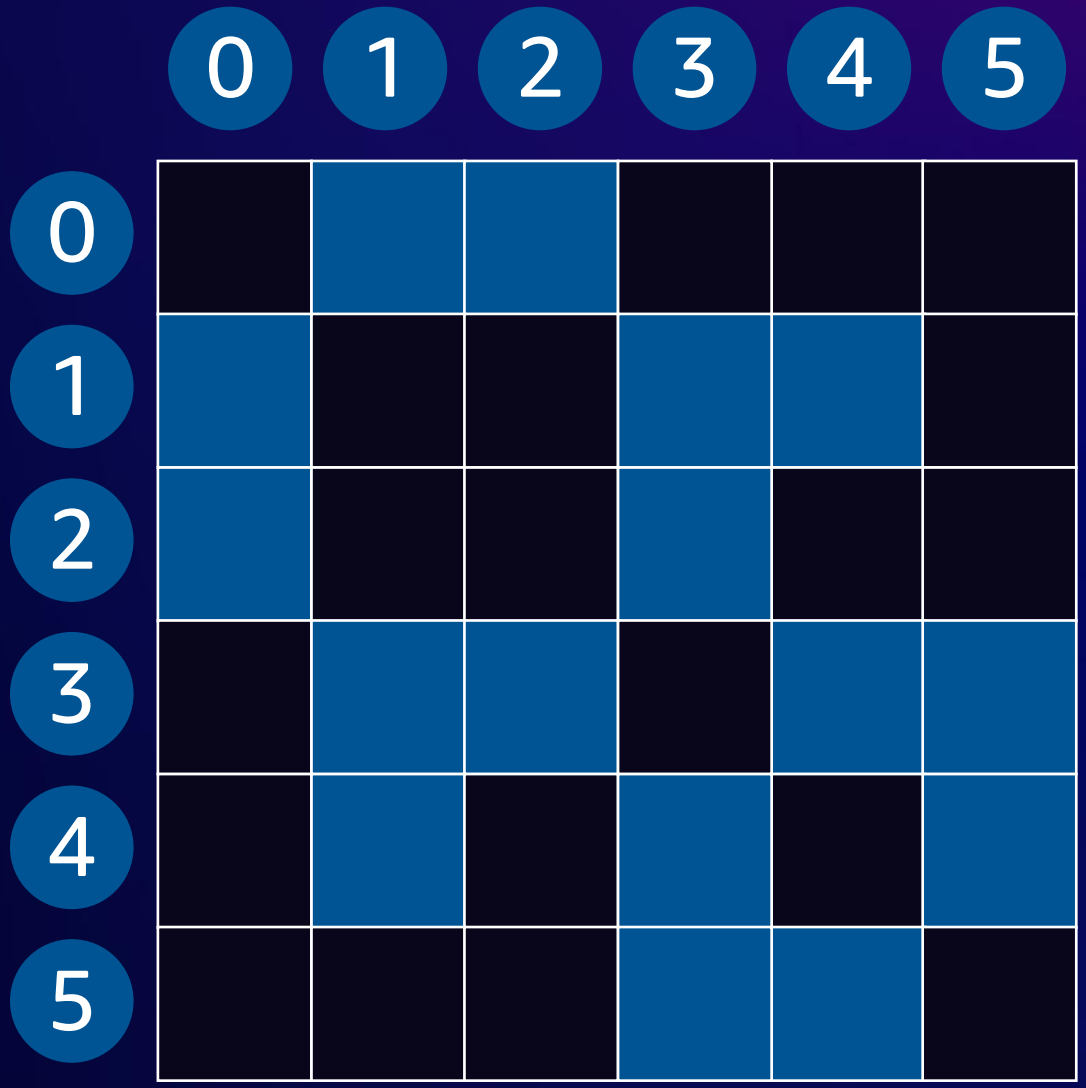
# How about the computational model

- Internally we provide
  - From “think like a vertex” to “think like a block (sub-graph)”
  - Visitor model

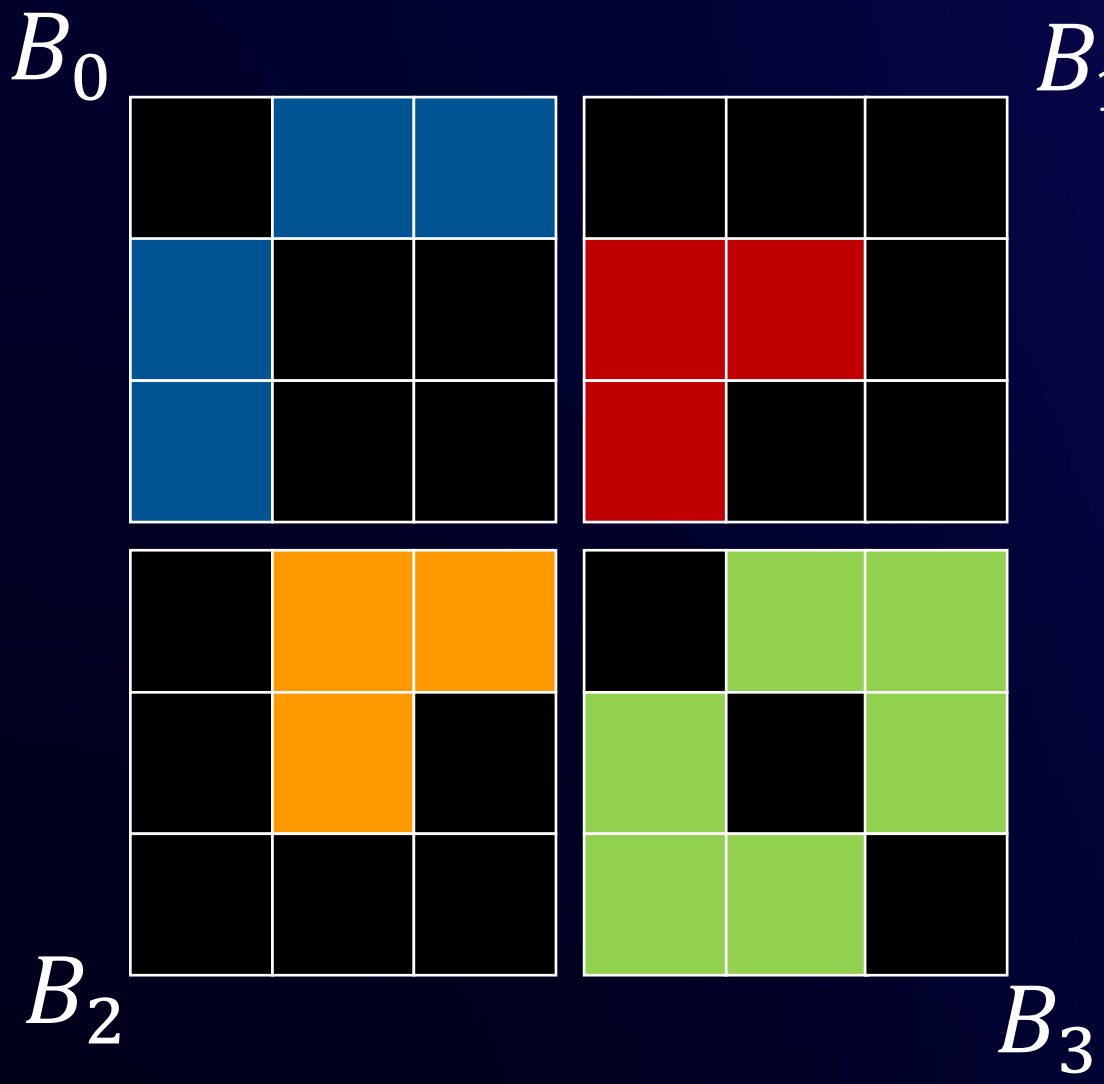
- Externally
  - Currently openCypher + with Graph API

# A toy graph and its matrix representation



# 2D “block” partitioning of edges




A block ($B_i$) : Set of edges

Graph, $G = \cup B_i$ , and $\cap B_i = \emptyset$

$B_0 = \{(0,1), (0,2), (1,0), (2,0)\}$

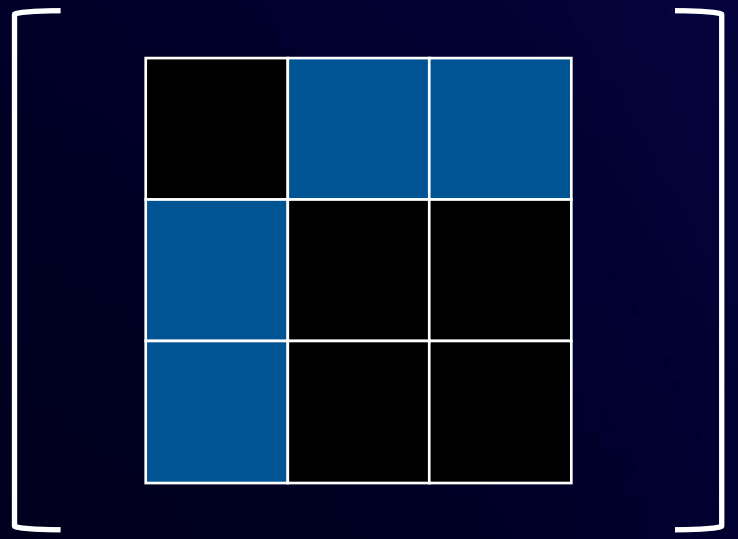
$B_1 = \{(1,3), (1,4), (2,3)\}$

$B_2 = \{(3,1), (3,2), (4,1)\}$

$B_3 = \{(3,4), (3,5), (4,3), (4,5), (5,3), (5,4)\}$

# Each algorithm is a one or more kernels

A kernel is functor that takes a block list as input

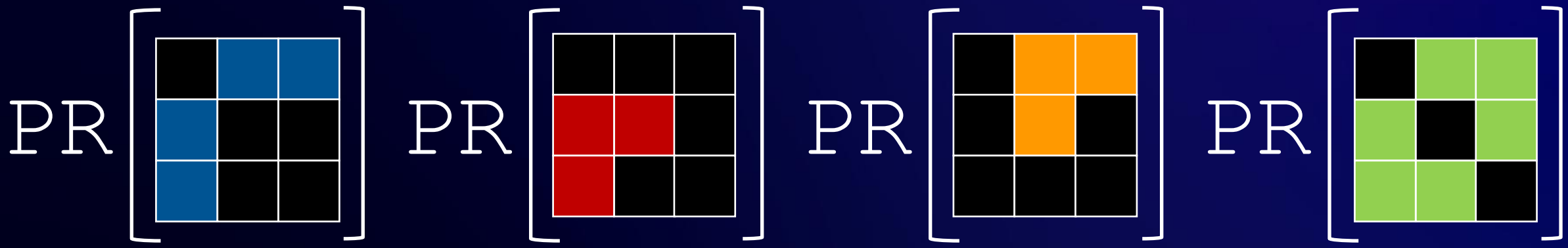
PR 

$$PageRank = \mathrm{U}_i \mathrm{PR}(\langle B_i \rangle)$$

Storage provides efficient index- and scan-based access/loops

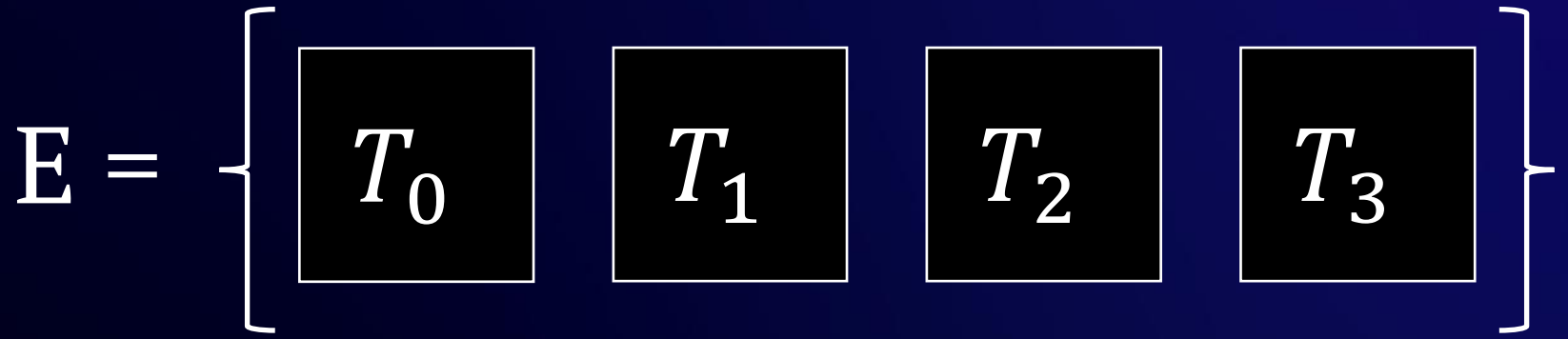
# Tasks

A task, $T_i$, is defined with a kernel that operates on a block list



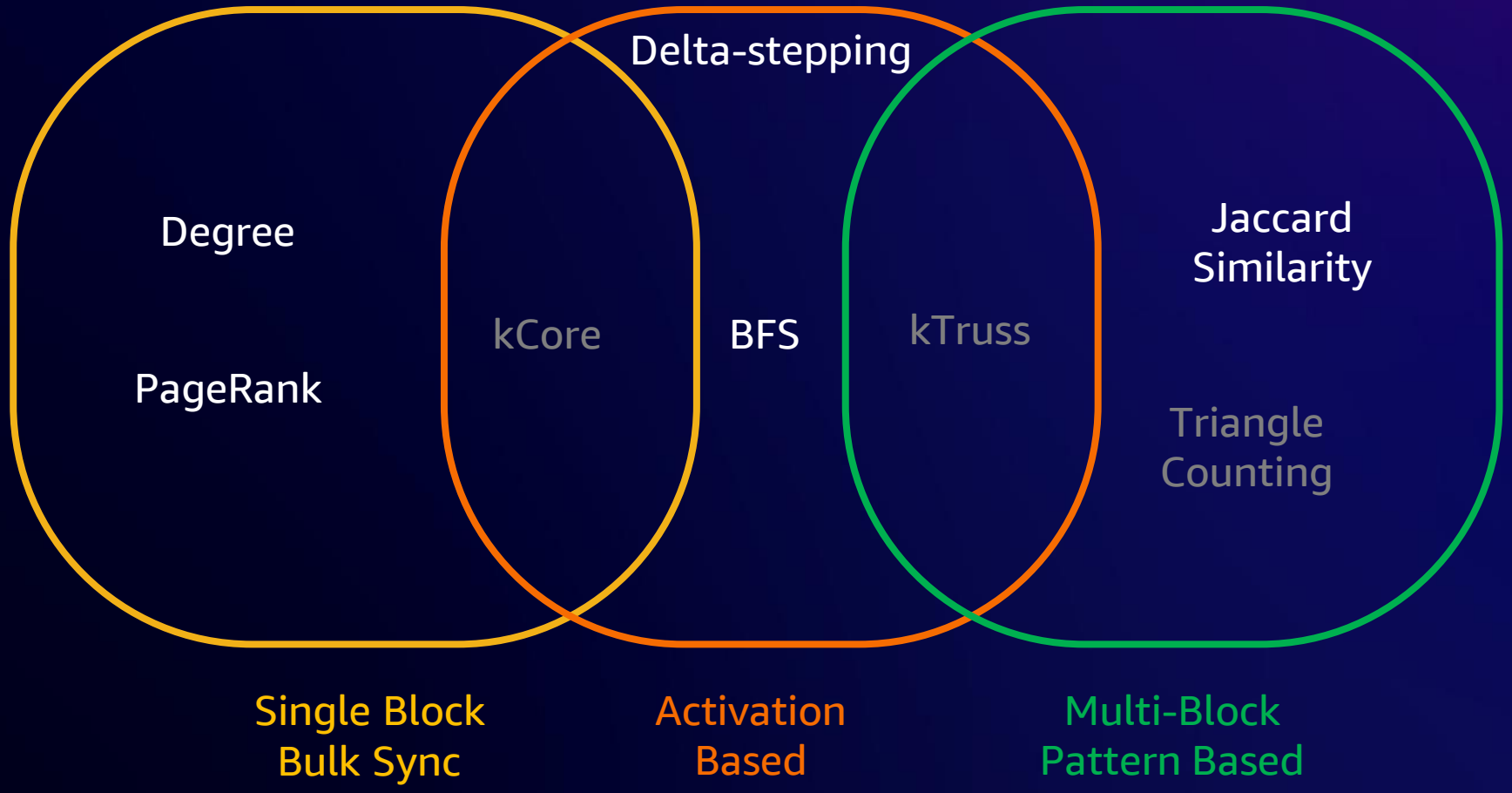
# Execution queue

Execution queue, E,  is a queue that contains tasks to compute




Amenable for heterogenous and distributed execution

# Categorizing graph algorithms



# Demo: Neptune Analytics

# Teaser: Graph interoperability

- What if we did not have to choose between RDF and LPG?
- What if we could use Gremlin over RDF, or SPARQL over LPG?
- Interoperability: single graph (meta)model, free use of any query language
- OneGraph (1G) model *"one graph to rule them all"*

Source: "Graph? Yes! Which one? Help!", O. Lassila, M. Schmidt, B. Bebee, D. Bechberger, W. Broekema, A. Khandelwal, K. Lawrence, R. Sharda, B. Thompson, arXiv:2110.13348v1, 2021.

# Thank you!

Please complete the session survey in the mobile app

**Brad Bebee**
beebs@amazon.com

**Ümit V. Çatalyürek**
uvc@amazon.com