



Let Me Graph that For You

Amazon Neptune Deep Dive & Workshop

Justin Thomas

Principal Business Development Manager



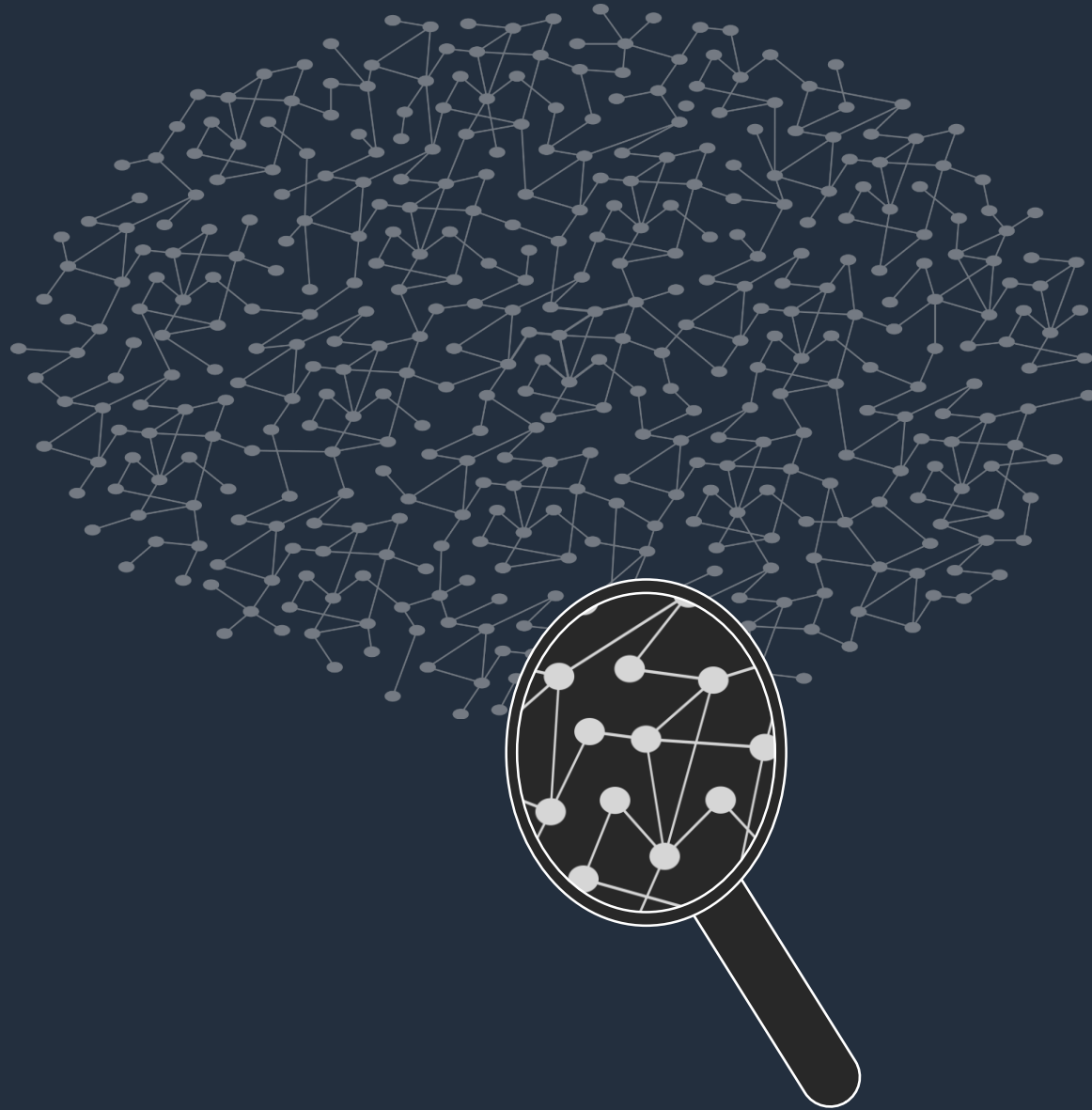
Agenda

- What is a Graph?
- Why use a Graph?
- Why use a Graph Database?
- Why use Amazon Neptune
- Demos
- Q&A

What is a Graph?



Graphs

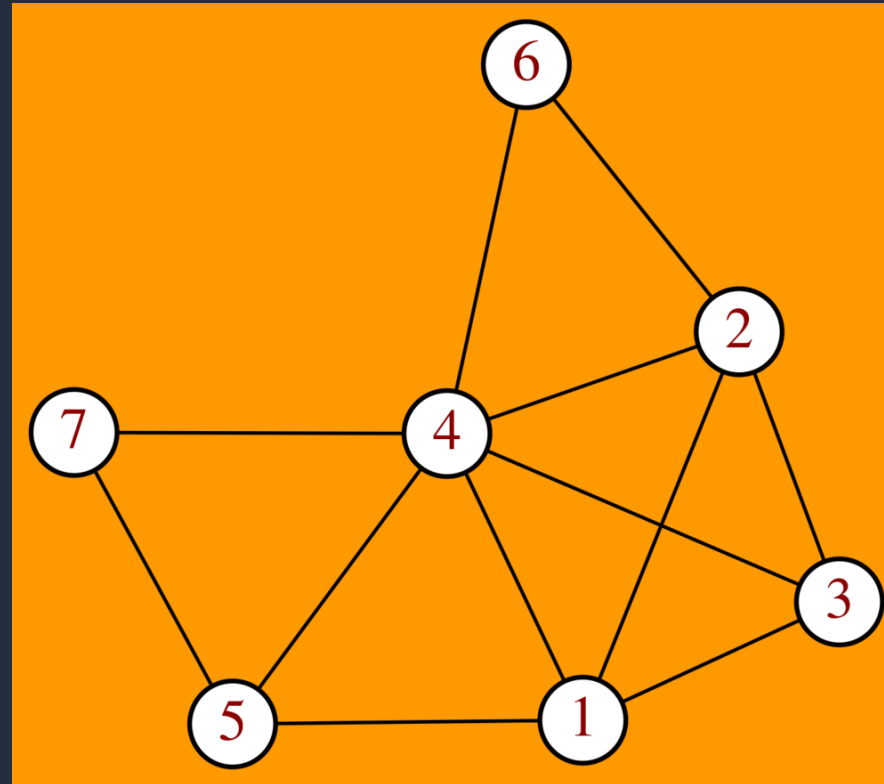


Model data based on relationships

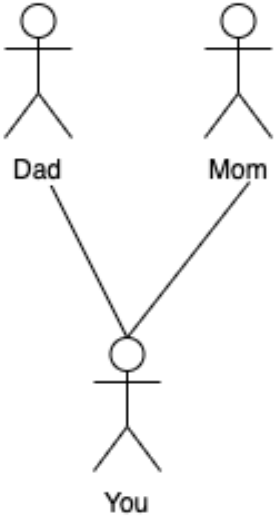
Explore connections and patterns
in connected data

Graph fundamentals

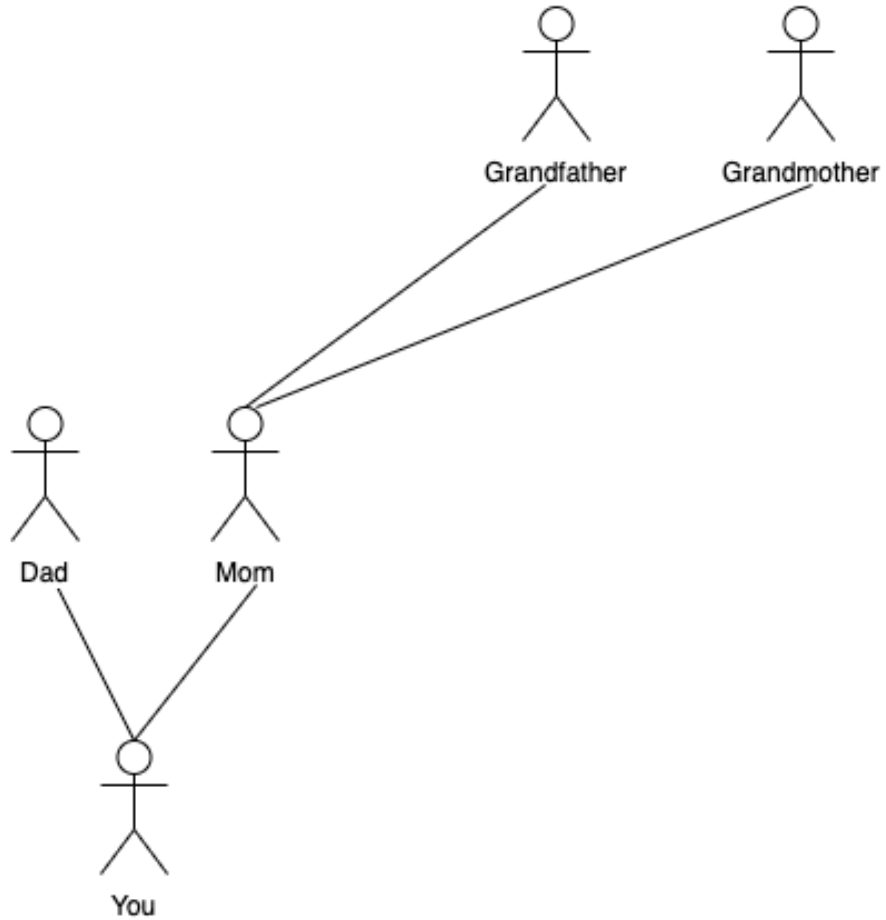
- Vertices or Nodes
- Edges or Relationships
- Edges can have direction



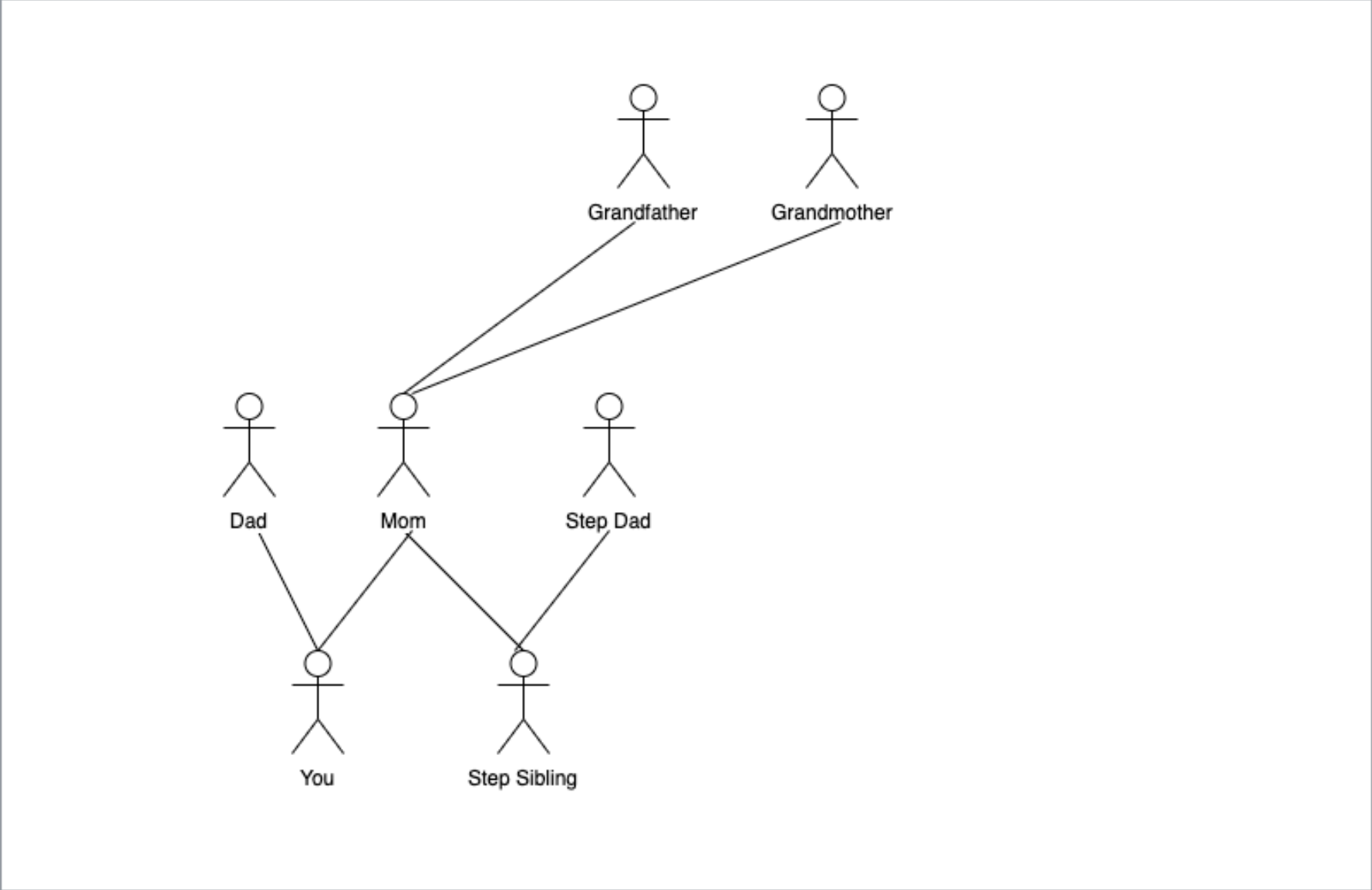
Graphs are intuitive



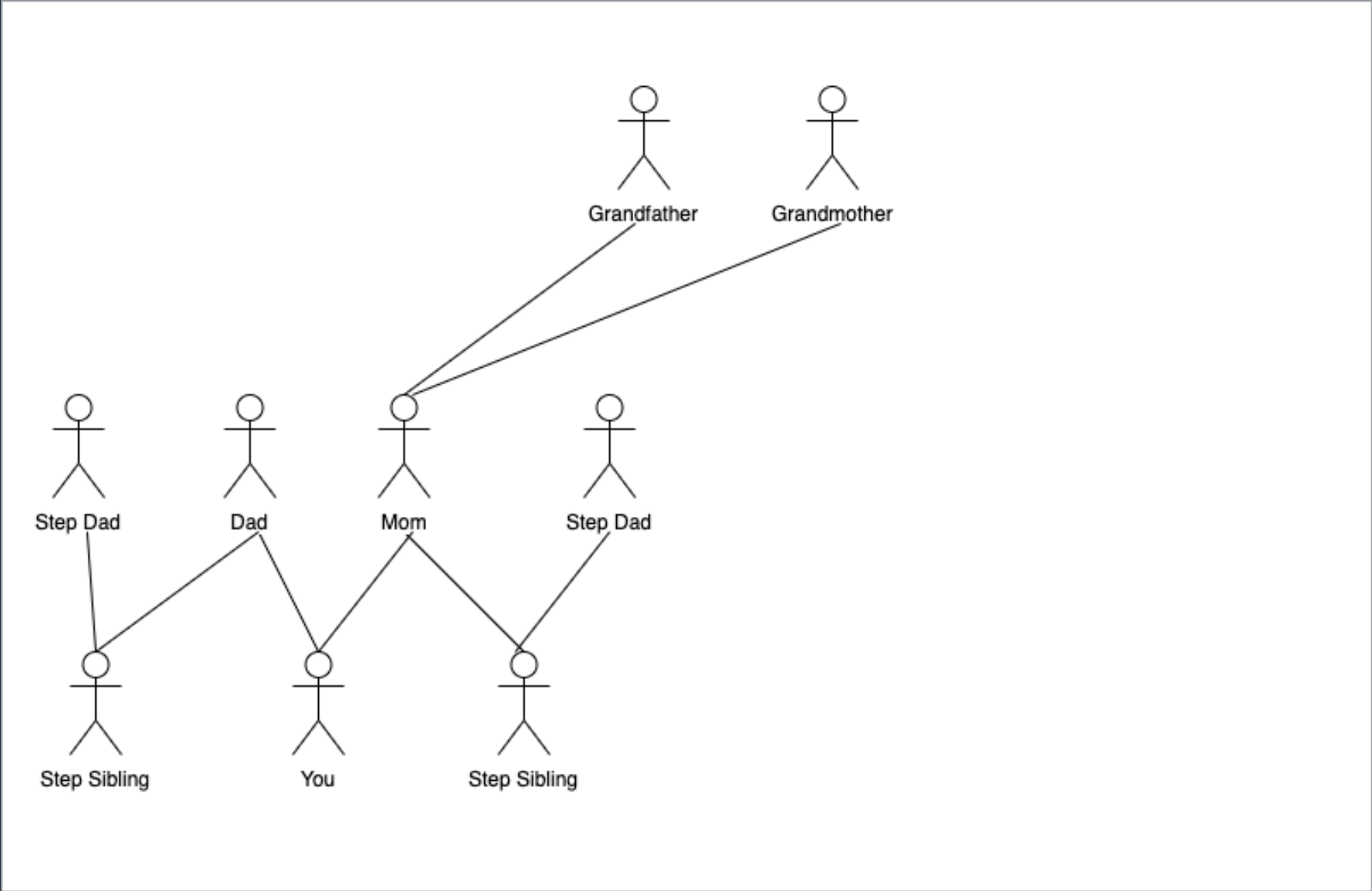
Graphs easily represent connections



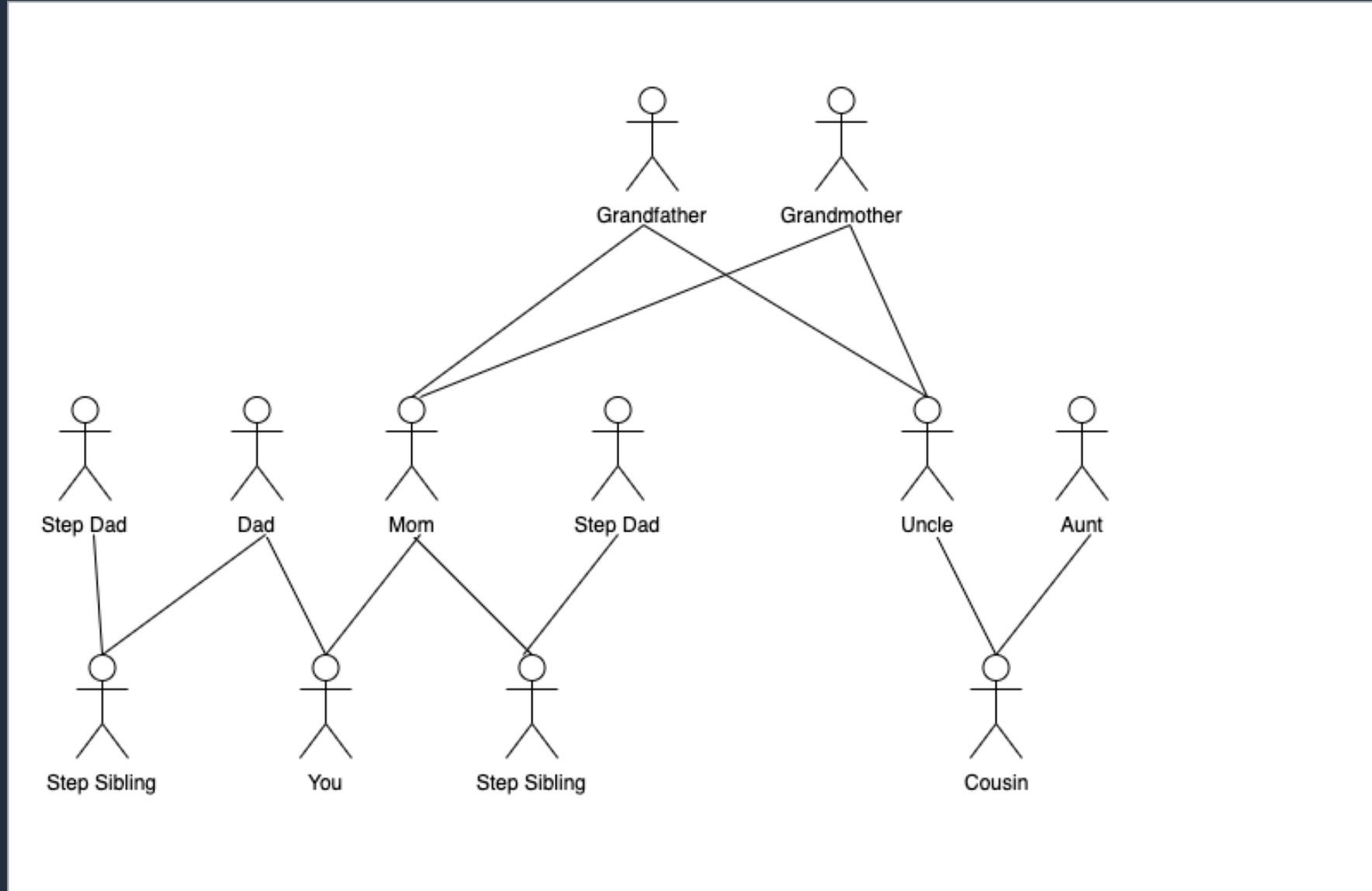
Sometimes the connections are complex



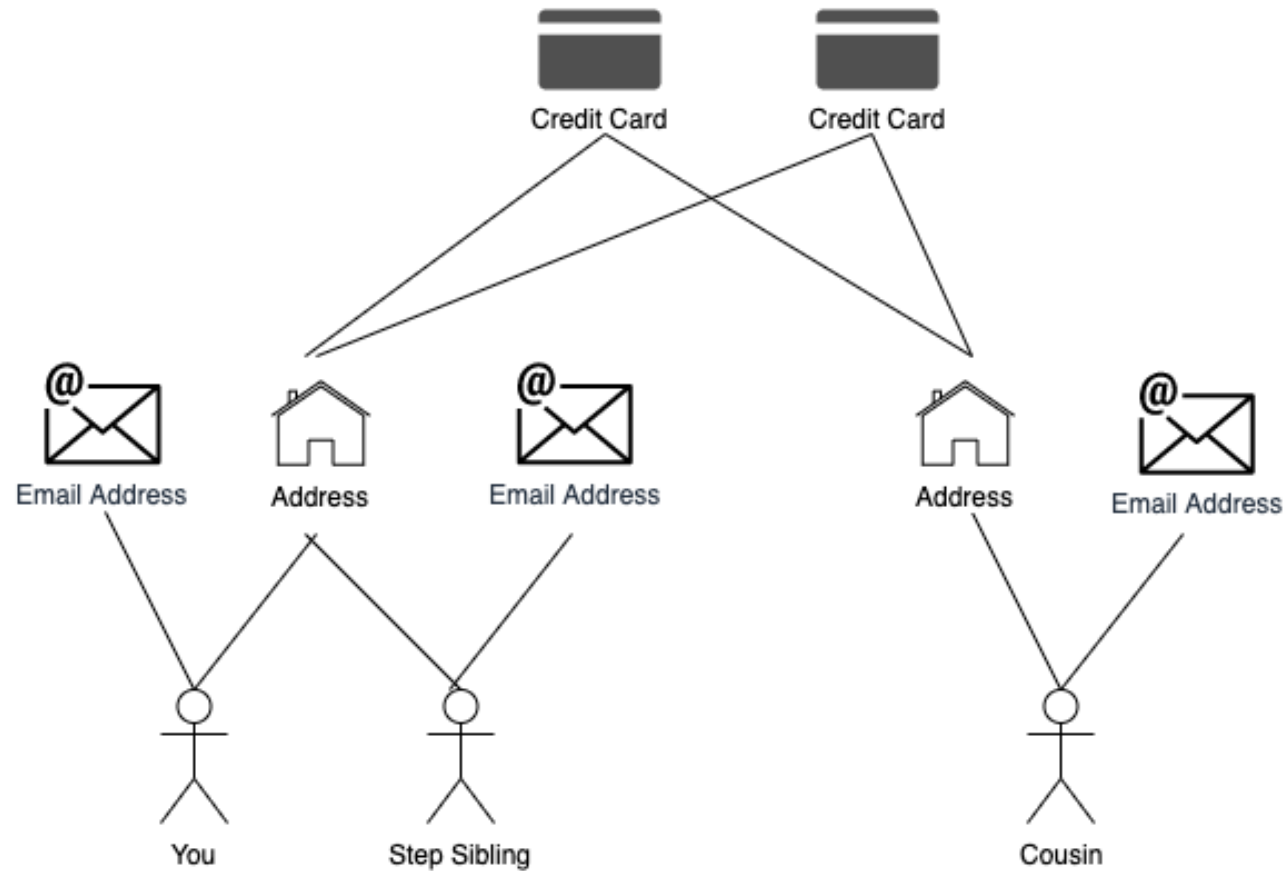
And varied



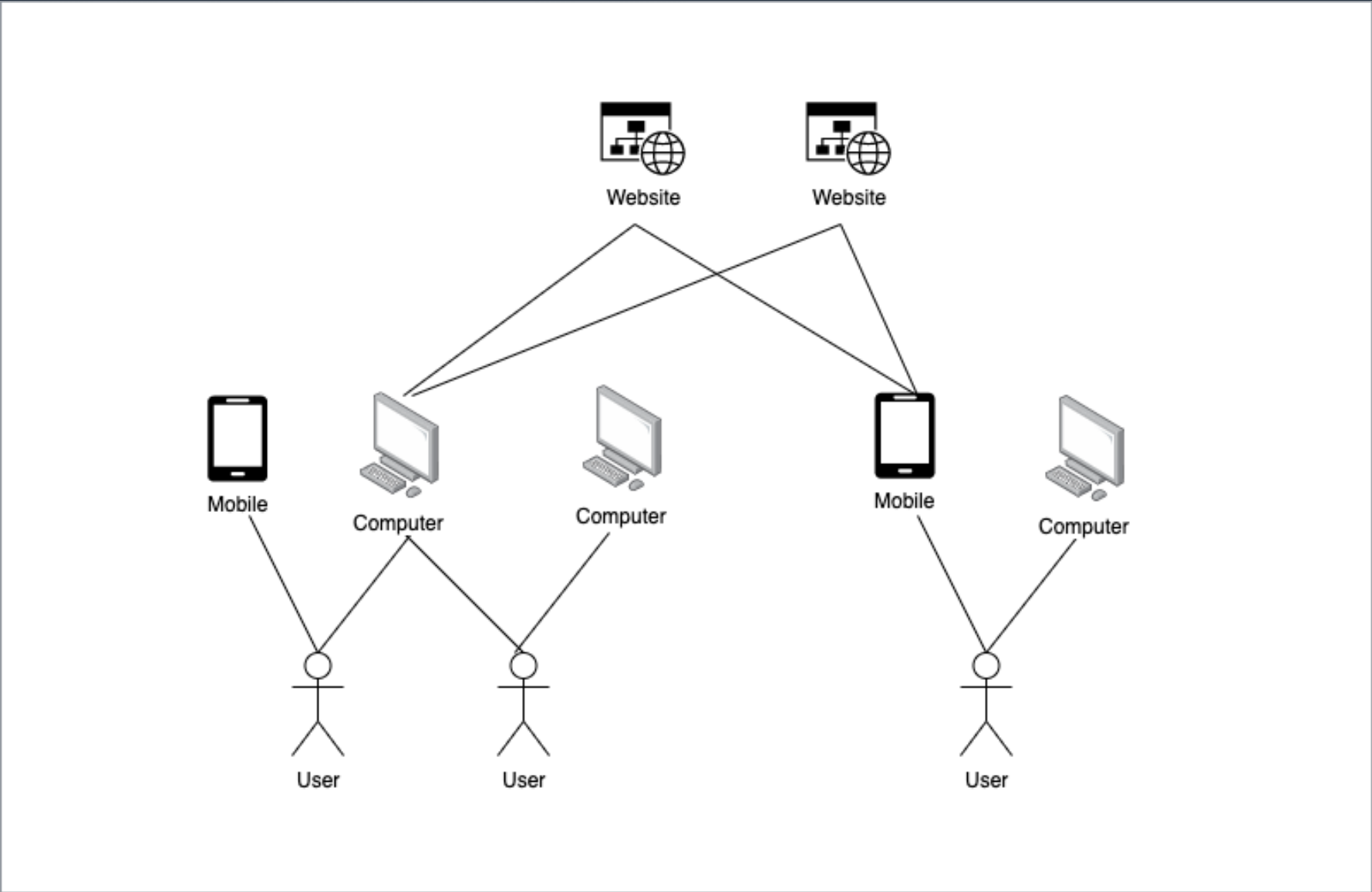
But they represent the world as it is



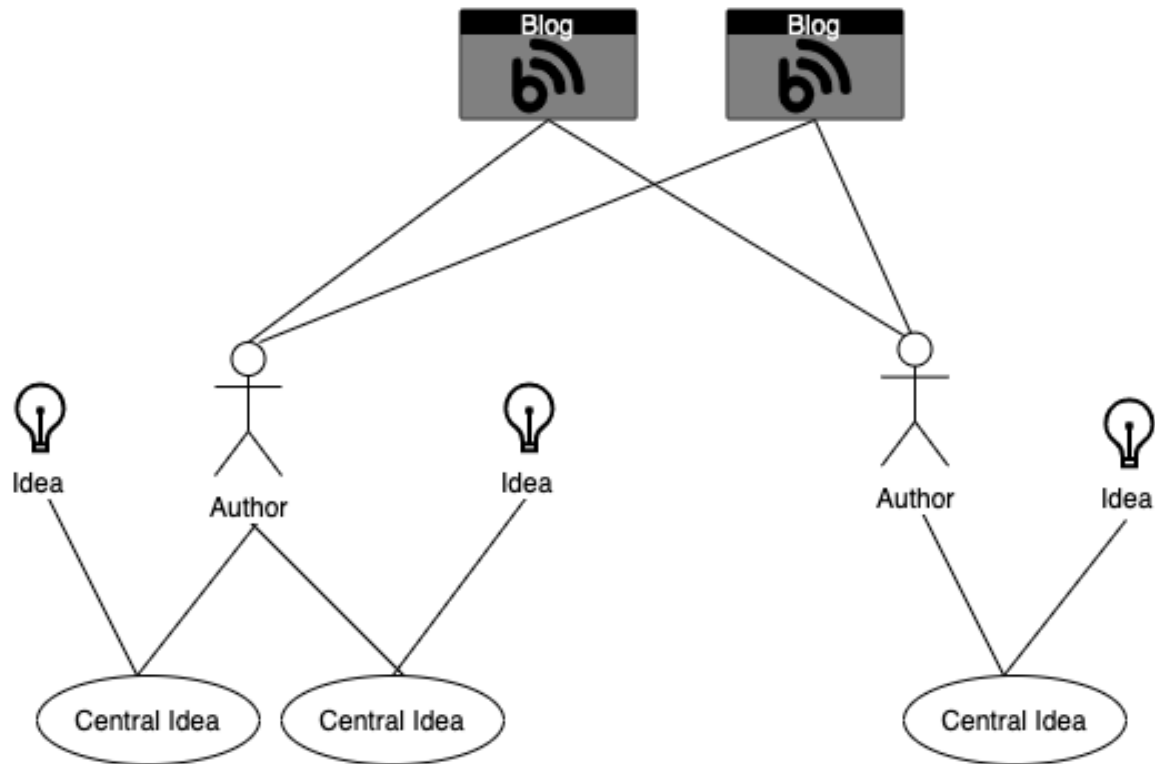
Graphs apply across domains, like Fraud



Or Identity Management



Or Knowledge Graphs



Technical challenges graphs help solve

- Combining data across data silos
- Finding common connections or paths
- Working with heterogenous data with complex relationships
- Data full of many-to-many relationships

It's the difference between using multiple excel spreadsheets or a mind map tool to think about your data.

Why use a Graph?



Common graph business problems

- We need to be better at detecting fraud
- My customers want better or more personalized recommendations
- We need to connect our siloed data sources
- We have multiple websites/applications and we need to link customer identities in these systems
- Our machine learning algorithms need improvement

Not so easily recognized graph problems

- Where are the risks in my IT Infrastructure/supply chain?
- Where did this data it come from?
- Why don't my search results relate to my question?
- How does person X have access to information Y?
- How will this price curve change impact my application?

Graphs solve the Where, Why, and How

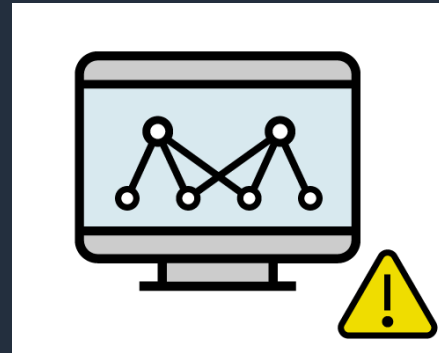
- **Where** are the risks in my IT Infrastructure/supply chain?
- **Where** did this data come from?
- **Why** don't my search results relate to my question?
- **How** does person X have access to information Y?
- **How** will this price curve change impact my application?

Why use a Graph Database?

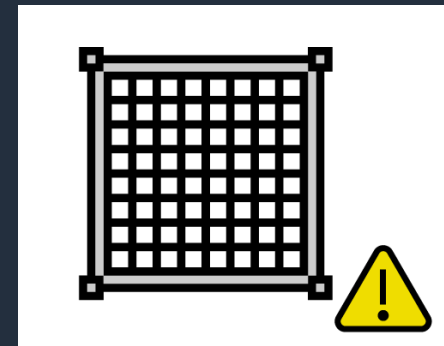
Challenges with highly connected data



Unnatural for
querying



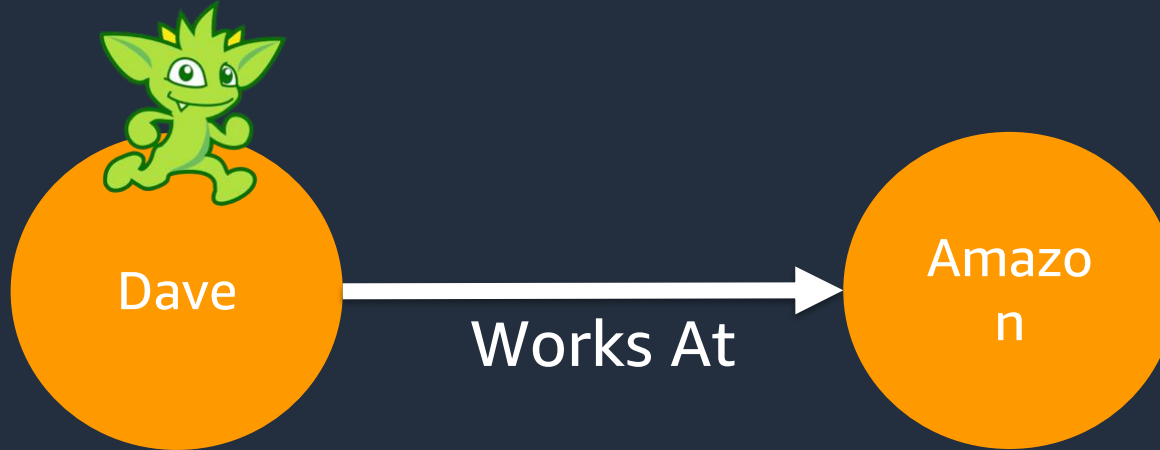
Inefficient
processing



Rigid schema inflexible
for changing data

Query languages- Designed to move through data

Graphs query languages are optimized to use connections to move through a network



Relational queries work by combining sets of data.



Efficient Processing

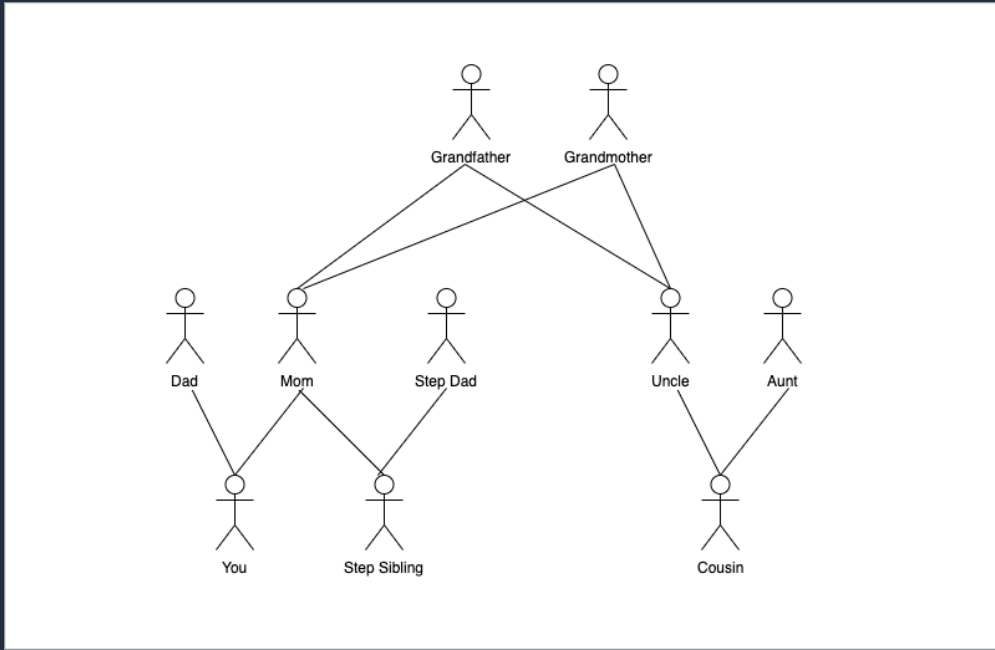
Let's look at the example:

Justin – Works At -> Amazon

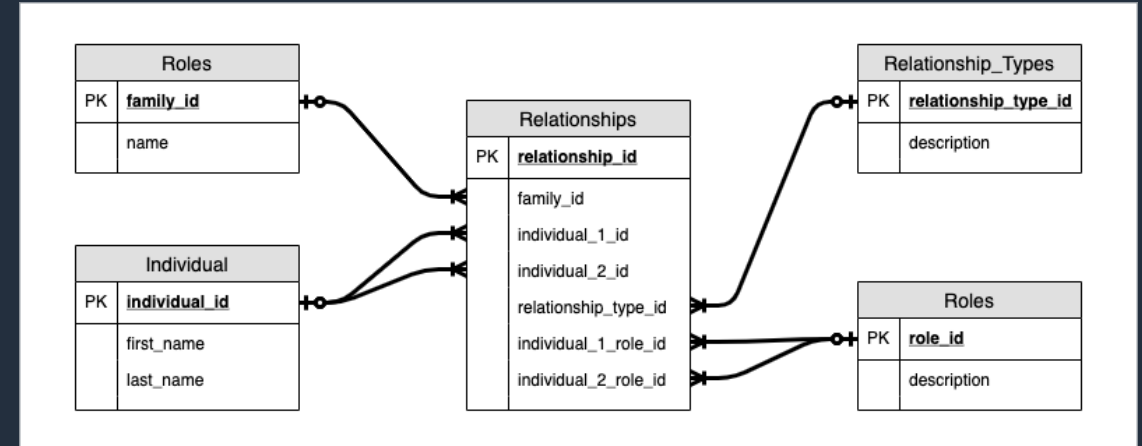
In a Graph database, the **Works At** connection is data, when needed the connection is retrieved

In a Relational databases, the **Works At** connection is metadata, when needed the connection must be calculated.

Schema flexibility makes adding new data easy



VS



Bonus - Graphs are easier to understand by new and/or non-technical people

Graph use cases are about the questions being asked

Graph databases excel at answering questions like:

- What does this person want to buy?
- How are these two people connected?
- Why did X impact Y?

These questions:

- Navigate (variably) connected structure
- Filter or compute a result on the basis of the *strength, weight, or quality* of relationships
- Require traversing an unknown number of connections

Other databases solve other questions better

Other databases excel answering to questions like, "How Much", "Count these items", "Find X based on Y"

- What were my sales for April 2021?
- How many people bought product X in Anchorage?
- Find me everyone named "Dave" in "Alaska"?
- Show me everyone who works at "Amazon"?
- Give me the value for key X?

Graph is not an Island - Purpose-built databases



Relational

Referential integrity, ACID transactions, schema-on-write



Key-value

High throughput, Low latency reads and writes, endless scale



Document

Store documents and quickly access querying on any attribute



In-memory

Query by key with microsecond latency



Graph

Quickly and easily create and navigate relationships between data



Time-series

Collect, store, and process data sequenced by time



Ledger

Complete, immutable, and verifiable history of all changes to application data



Wide Column

Scalable, highly available, and managed Apache Cassandra-compatible service

AWS Service(s)



Aurora RDS



DynamoDB



DocumentDB



ElastiCache



Neptune



Timestream



QLDB



Keyspaces
Managed Cassandra

Common Use Cases

Lift and shift, ERP, CRM, finance

Real-time bidding, shopping cart, social, product catalog, customer preferences

Content management, personalization, mobile

Leaderboards, real-time analytics, caching

Fraud detection, social networking, recommendation engine

IoT applications, event tracking

Systems of record, supply chain, health care, registrations, financial

Build low-latency applications, leverage open source, migrate Cassandra to the cloud

Graphs and other databases

Graph databases are frequently used in conjunction with other databases

- Other databases are often used to answer questions where there is a single distinct answer.
 - Whom bought this product?
 - What is the average rating of this movie?
- Graph are good at giving you answers that a range of answers.
 - What product is this person going to buy next?
 - What movie will this person like?

Why use Amazon Neptune?

Amazon Neptune

Fast, reliable graph database built for the cloud

Open



Supports Apache TinkerPop & W3C RDF graph models

Fast



Query billions of relationships with millisecond latency

Reliable



6 replicas of your data across 3 AZs with full backup and restore

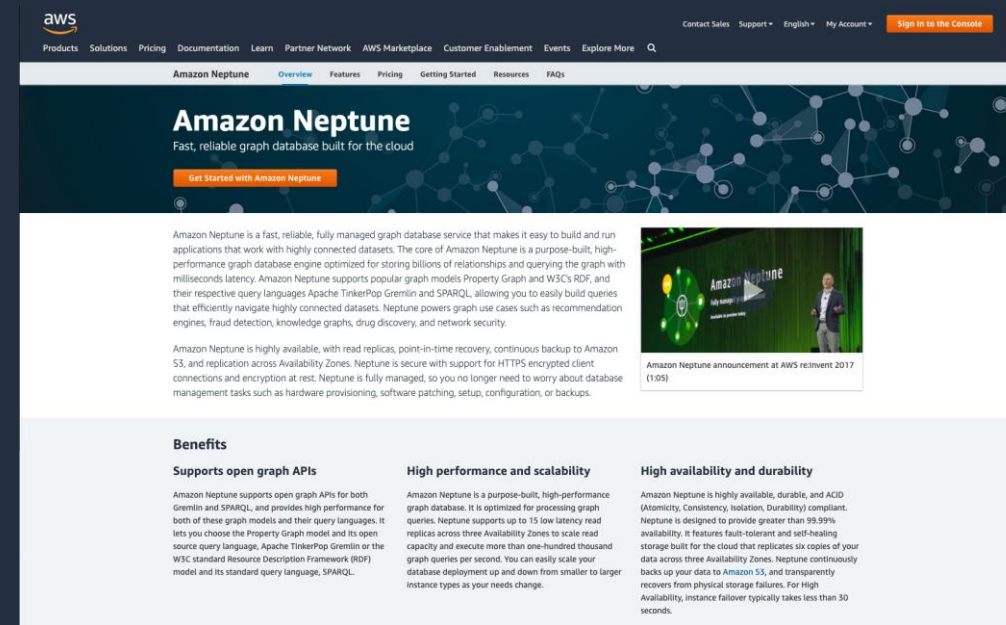
Easy



Build powerful queries easily with Gremlin and SPARQL

Neptune General Availability

- Announced on 5/30/2018
- 21 AWS Regions: US East (N. Virginia, Ohio), US West (Oregon, N. California), Canada (Central), Europe (Ireland, London, Frankfurt, Paris, Stockholm), Middle East (Bahrain), Asia Pacific (Mumbai, Singapore, Sydney, Seoul, Tokyo), China (Ningxia), GovCloud (East, West)
- ISO, HIPAA, SOC, PCI/DSS Compliance Certifications

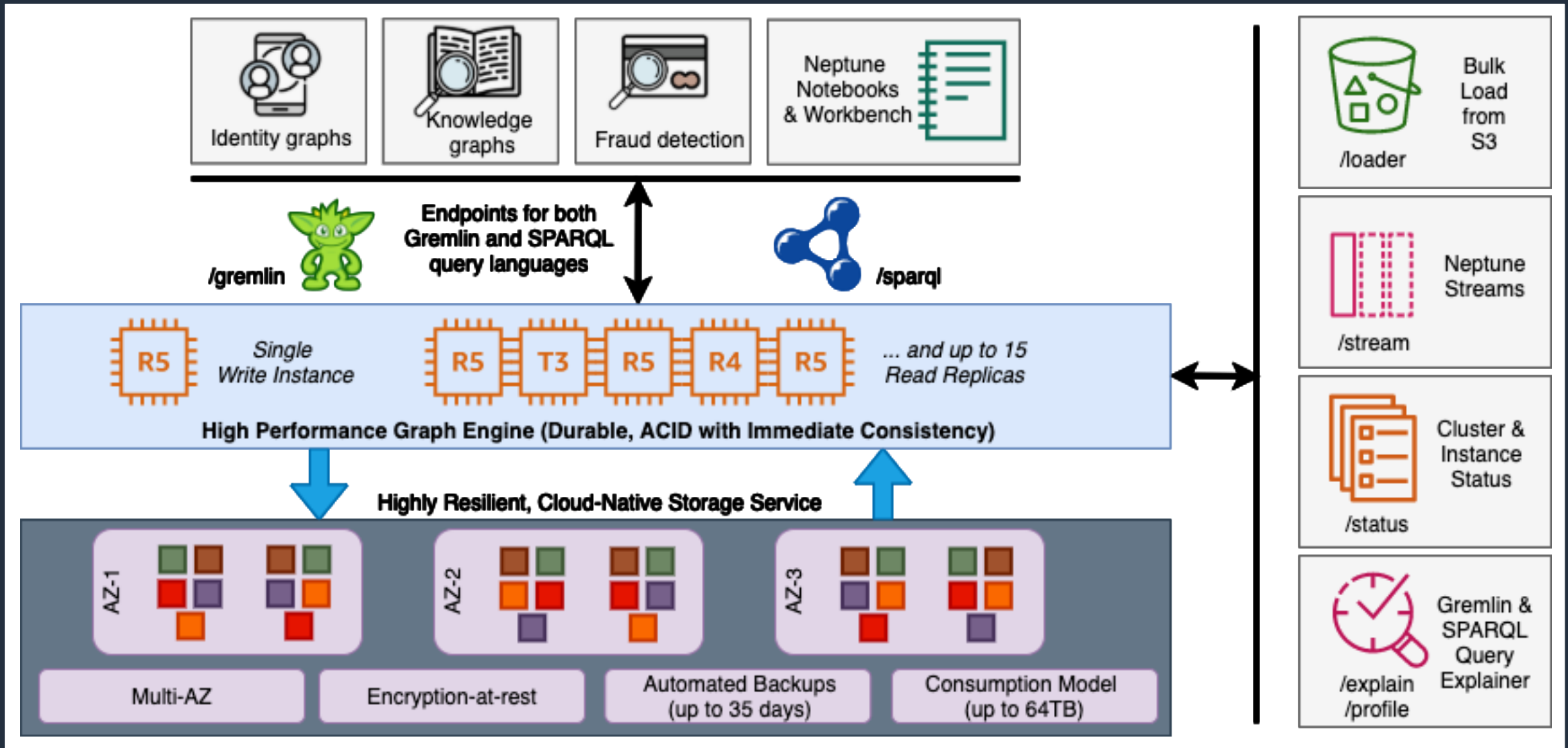


The screenshot shows the Amazon Neptune product page. At the top, there's a navigation bar with the AWS logo and links for Products, Solutions, Pricing, Documentation, Learn, Partner Network, AWS Marketplace, Customer Enablement, Events, and Explore More. A search icon is also present. Below the navigation bar, the main heading reads "Amazon Neptune" with the tagline "Fast, reliable graph database built for the cloud". A "Get Started with Amazon Neptune" button is visible. The main content area features a large image of a person presenting at a conference, with the text "Amazon Neptune announcement at AWS re:invent 2017 (1:05)". Below this, there are three columns of text under the heading "Benefits".

Benefits

- Supports open graph APIs**
Amazon Neptune supports open graph APIs for both Gremlin and SPARQL, and provides high performance for both of these graph models and their query languages. It lets you choose the Property Graph model and its open source query language, Apache TinkerPop Gremlin or the W3C standard Resource Description Framework (RDF) model and its standard query language, SPARQL.
- High performance and scalability**
Amazon Neptune is a purpose-built, high-performance graph database. It is optimized for processing graph queries. Neptune supports up to 15 low latency read replicas across three Availability Zones to scale read capacity and execute more than one-hundred thousand graph queries per second. You can easily scale your database deployment up and down from smaller to larger instance types as your needs change.
- High availability and durability**
Amazon Neptune is highly available, durable, and ACID (Atomicity, Consistency, Isolation, Durability) compliant. Neptune is designed to provide greater than 99.99% availability. It features fault-tolerant and self-healing storage built for the cloud that replicates six copies of your data across three Availability Zones. Neptune continuously backs up your data to Amazon S3, and transparently recovers from physical storage failures. For High Availability, instance failover typically takes less than 30 seconds.

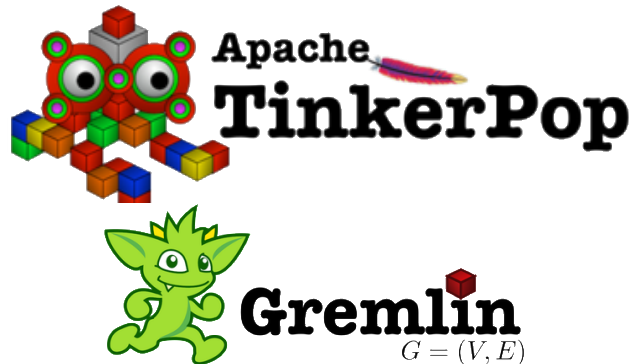
Amazon Neptune High Level Architecture



Leading graph models and frameworks

PROPERTY GRAPH

Open Source Apache TinkerPop™
Gremlin Traversal Language
Programming language drivers



RESOURCE DESCRIPTION FRAMEWORK (RDF)

W3C Open Standard
SPARQL Query Language



Common Neptune workloads

Social networks



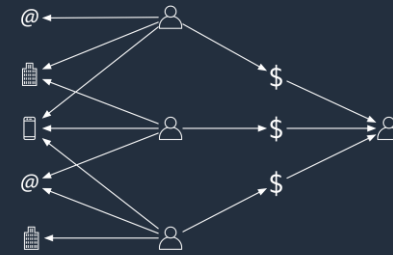
Social **connections** between friends and colleagues

Identity graphs



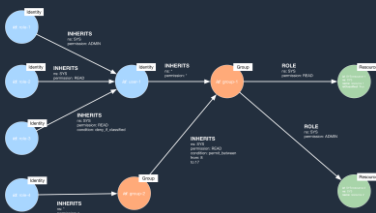
Unified view by **linking** devices, IP addresses, & browsing behaviors

Fraud detection



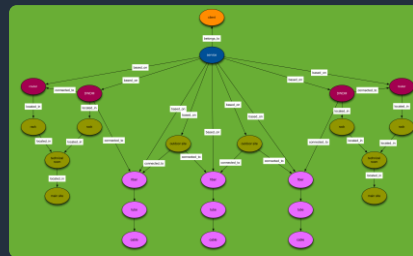
Patterns of connected transactions, senders, receivers, and identity info

Entitlements



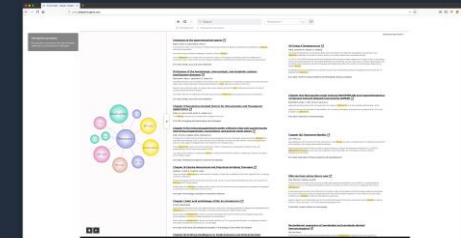
Conditional **path** from identities to resources

Network analysis



Top-down and bottom-up **parent-child** traversals

Knowledge graphs



Context created by **linked** datasets and concepts

Demos



Resources



Documentation

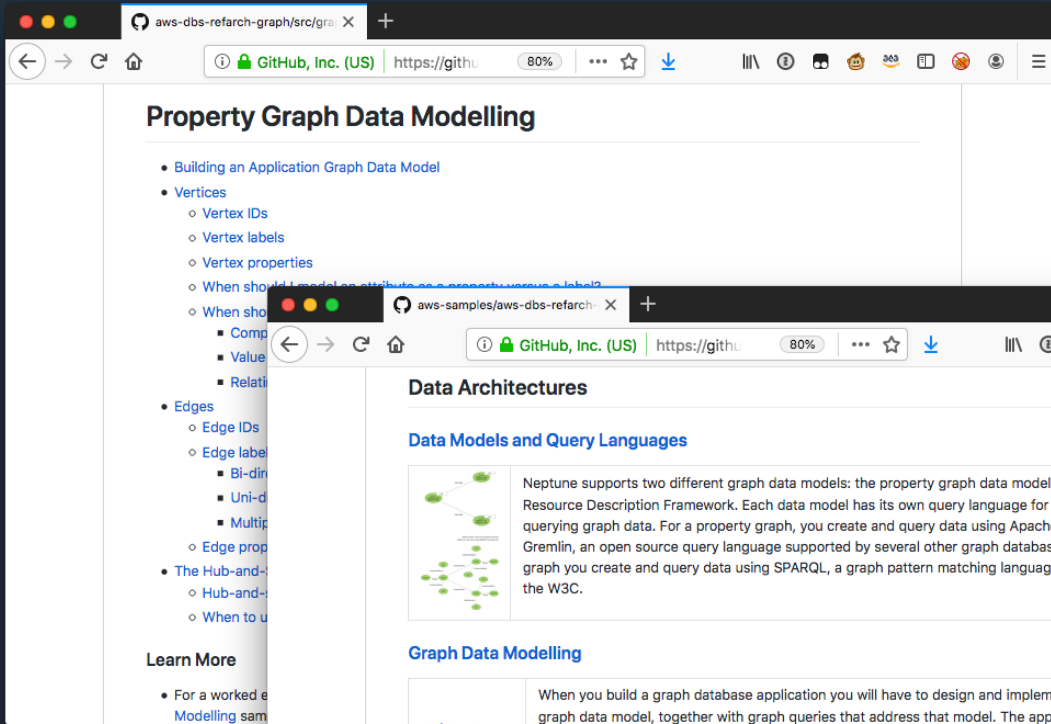
Start with the 'What is Neptune?' and 'Neptune Overview' sections

The screenshot shows the AWS Neptune documentation page. The breadcrumb trail is "AWS Documentation » Neptune » User Guide » Overview of Amazon Neptune Features". A notification box at the top states: "The AWS Documentation website is getting a new look! Try it now and let us know what you think. Switch to the new look >>". Below this, a message says: "You can return to the original look by selecting English in the language selector above." The main heading is "Overview of Amazon Neptune Features". The text reads: "This section provides an overview of Neptune features, including clusters, instances, and storage characteristics of Neptune graphs." A "Note" section follows: "This section does not cover access to the data in a Neptune graph. For information about how to connect to a running Neptune DB cluster with Gremlin, see [Accessing the Neptune Graph with Gremlin](#). For information about how to connect to a running Neptune DB cluster with SPARQL, see [Accessing the Neptune Graph with SPARQL](#)." A "Topics" section lists: "What Is a Graph Database?", "Amazon Neptune DB Clusters", "Amazon Neptune Graph Data Model", "Amazon Neptune Storage", "Amazon Neptune Reliability", "High Availability for Neptune", "Connecting to Amazon Neptune Endpoints", "Replication with Amazon Neptune", and "Changes and Updates to Amazon Neptune". The left sidebar contains a navigation menu with "Neptune Overview" selected, and other items like "DB Clusters", "Graph Data Model", "Storage", "Reliability", "High Availability", "Endpoint Connections", "Neptune Replication", "Latest Updates", "Security", "Getting Started", "Gremlin", "SPARQL", "Loading Data into Neptune", "Managing Neptune on the Console", "Backing Up and Restoring", "Monitoring Neptune", "Best Practices", "Neptune Limits", and "Neptune Errors".

The screenshot shows the AWS Neptune documentation page for "Using an AWS CloudFormation Stack to Create a Neptune DB Cluster". The breadcrumb trail is "AWS Documentation » Neptune » User Guide » Using an AWS CloudFormation Stack to Create a Neptune DB Cluster". The text reads: "You can use an AWS CloudFormation template to set up a Neptune DB Cluster." A numbered list starts with: "1. To launch the AWS CloudFormation stack on the AWS CloudFormation console, choose one of the **Launch Stack** buttons in the following table." Below this is a table with columns: "Region", "View", "View in Designer", and "Launch". The table lists various regions with corresponding "View" and "View in Designer" links and "Launch Stack" buttons. The "Launch" column contains "Launch Stack" buttons with a dropdown arrow. The regions listed are: US East (N. Virginia), US East (Ohio), US West (Oregon), EU (Ireland), EU (London), EU (Frankfurt), Asia Pacific (Singapore), Asia Pacific (Sydney), Asia Pacific (Tokyo), Asia Pacific (Mumbai), Asia Pacific (Seoul), EU (Stockholm), AWS GovCloud (US-West), and AWS GovCloud (US-East). Below the table, the list continues with: "2. On the **Select Template** page, choose **Next**." and "3. On the **Specify Details** page, choose a key pair for the **EC2SSHPairName**." A note follows: "This key pair is required to access the EC2 instance. Ensure that you have the PEM file for". The left sidebar is identical to the previous screenshot, with "Neptune Overview" selected.

Reference architectures

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

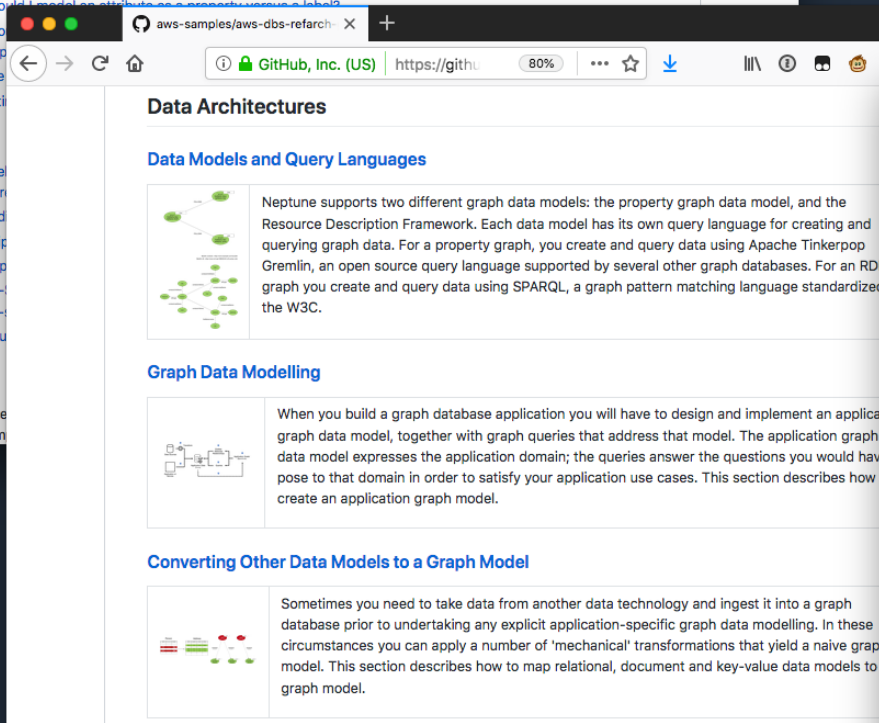


Property Graph Data Modelling

- Building an Application Graph Data Model
- Vertices
 - Vertex IDs
 - Vertex labels
 - Vertex properties
 - When should I model an attribute as a property versus a label?
 - When should I model an attribute as a property versus a label?
- Edges
 - Edge IDs
 - Edge labels
 - Bi-directional
 - Unidirectional
 - Multiplicity
 - Edge properties
- The Hub-and-Spoke Model
 - Hub-and-Spoke
 - When to use

Learn More

- For a worked example, see [Modelling sample](#)



Data Architectures

Data Models and Query Languages

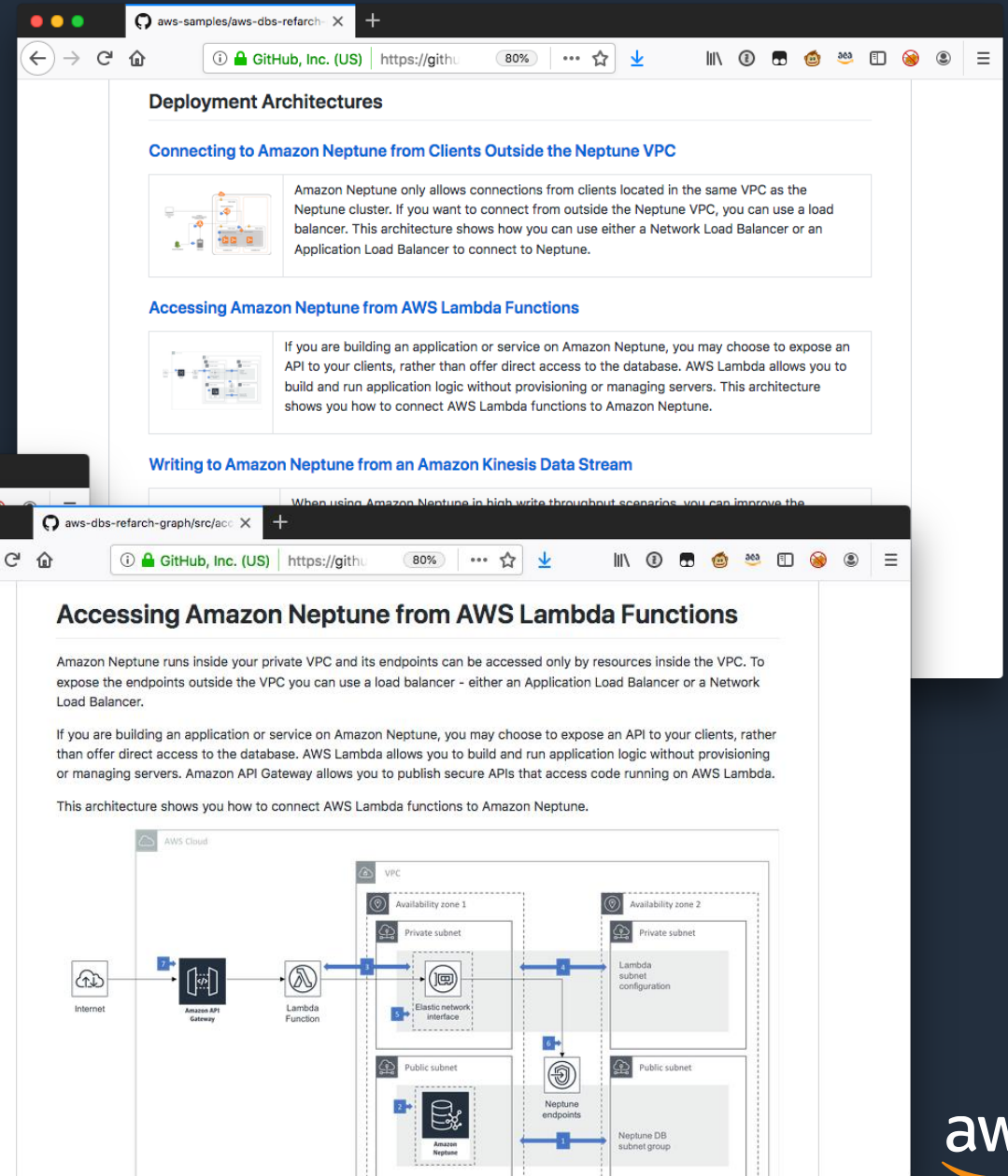
Neptune supports two different graph data models: the property graph data model, and the Resource Description Framework. Each data model has its own query language for creating and querying graph data. For a property graph, you create and query data using Apache Tinkerpop Gremlin, an open source query language supported by several other graph databases. For an RDF graph you create and query data using SPARQL, a graph pattern matching language standardized the W3C.

Graph Data Modelling

When you build a graph database application you will have to design and implement an application graph data model, together with graph queries that address that model. The application graph data model expresses the application domain; the queries answer the questions you would have posed to that domain in order to satisfy your application use cases. This section describes how to create an application graph model.

Converting Other Data Models to a Graph Model

Sometimes you need to take data from another data technology and ingest it into a graph database prior to undertaking any explicit application-specific graph data modelling. In these circumstances you can apply a number of 'mechanical' transformations that yield a naive graph model. This section describes how to map relational, document and key-value data models to a graph model.



Deployment Architectures

Connecting to Amazon Neptune from Clients Outside the Neptune VPC

Amazon Neptune only allows connections from clients located in the same VPC as the Neptune cluster. If you want to connect from outside the Neptune VPC, you can use a load balancer. This architecture shows how you can use either a Network Load Balancer or an Application Load Balancer to connect to Neptune.

Accessing Amazon Neptune from AWS Lambda Functions

If you are building an application or service on Amazon Neptune, you may choose to expose an API to your clients, rather than offer direct access to the database. AWS Lambda allows you to build and run application logic without provisioning or managing servers. This architecture shows you how to connect AWS Lambda functions to Amazon Neptune.

Writing to Amazon Neptune from an Amazon Kinesis Data Stream

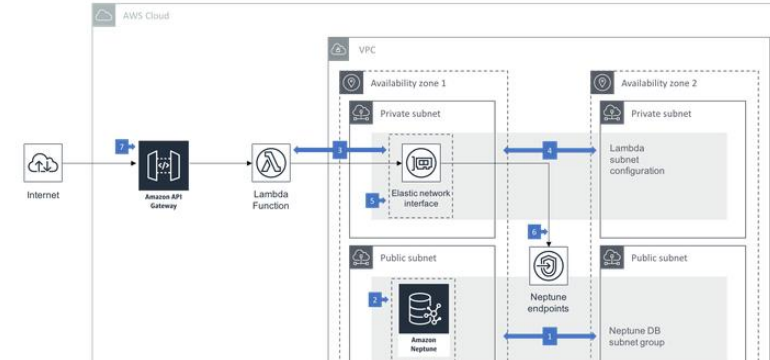
When using Amazon Neptune in high write throughput scenarios, you can improve the performance of your application by using Amazon Kinesis Data Streams to ingest data into Neptune.

Accessing Amazon Neptune from AWS Lambda Functions

Amazon Neptune runs inside your private VPC and its endpoints can be accessed only by resources inside the VPC. To expose the endpoints outside the VPC you can use a load balancer - either an Application Load Balancer or a Network Load Balancer.

If you are building an application or service on Amazon Neptune, you may choose to expose an API to your clients, rather than offer direct access to the database. AWS Lambda allows you to build and run application logic without provisioning or managing servers. Amazon API Gateway allows you to publish secure APIs that access code running on AWS Lambda.

This architecture shows you how to connect AWS Lambda functions to Amazon Neptune.



The diagram illustrates a VPC architecture for accessing Amazon Neptune from AWS Lambda functions. It shows the Internet, Amazon API Gateway, Lambda Function, Elastic network interface, Neptune endpoints, and Neptune DB subnet group within a VPC. The VPC is divided into two availability zones, each with private and public subnets. The Lambda Function is connected to the Elastic network interface, which is connected to the Neptune endpoints. The Neptune endpoints are connected to the Neptune DB subnet group.

Samples

<https://github.com/aws-samples/amazon-neptune-samples>

amazon-neptune-samples/gremlin

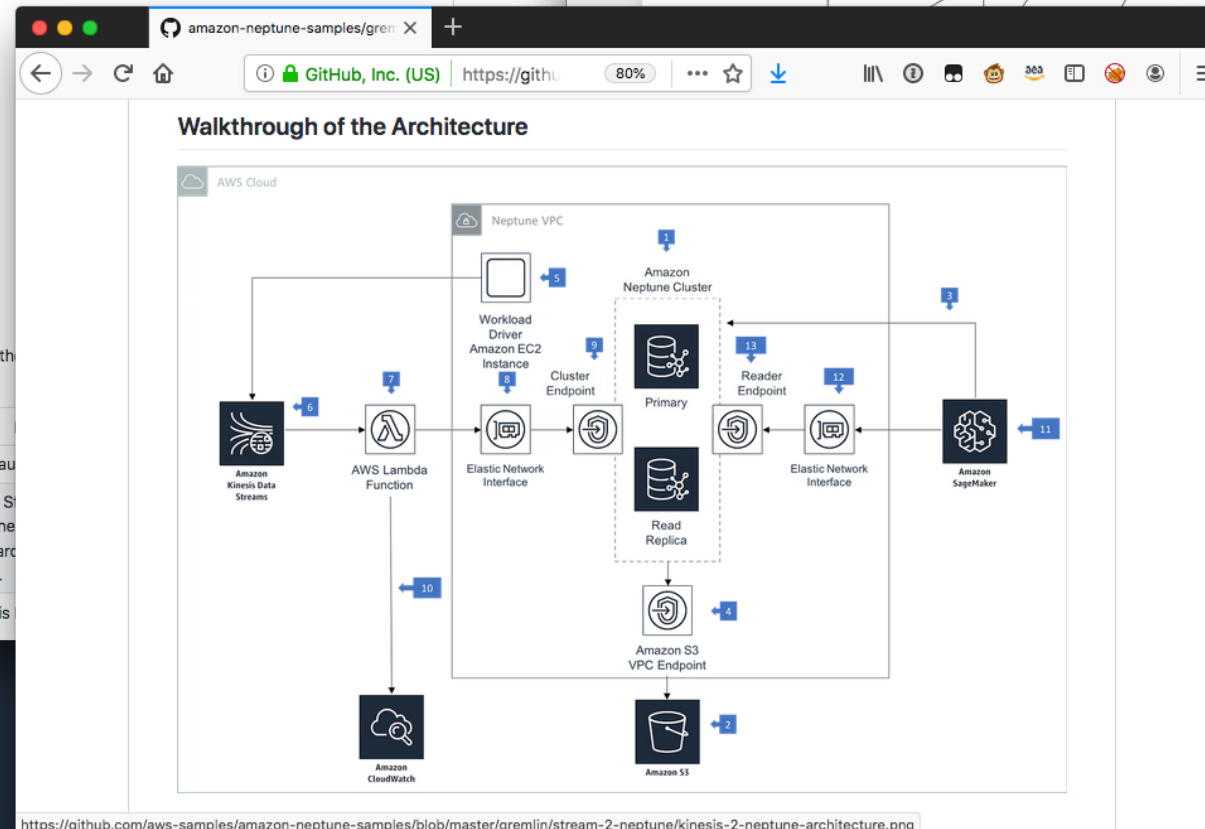
Setup

Install the components using CloudFormation:

Region	Stack
US East (N. Virginia)	Launch Stack
US East (Ohio)	Launch Stack
US West (Oregon)	Launch Stack
EU (Ireland)	Launch Stack
EU (London)	Launch Stack
EU (Frankfurt)	Launch Stack

You'll need to supply an `EC2KeyPairName` so that you can access the instance. You also specify a number of other parameters:

Parameter	Default Value
<code>DbInstanceType</code>	Neptune database instance type (default is <code>db.neptune4.xlarge</code>)
<code>ShardCount</code>	Number of shards in the Kinesis Data Stream (the number of vCPUs on the Neptune instance create 64 shards). The number of shards must be a multiple of the number of shards per function writing batches to Neptune.
<code>BatchReadSize</code>	Number of records read from a Kinesis Data Stream writes to Neptune (default 1000)



Neptune/Getting-Started/ data-model-3

https://neptunenotebookinstan 90%

Jupyter data-model-3

Last Checkpoint: a few seconds ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3

Data model

The graph shows relationships between roles (Manager, Associate Analyst, Any Company, Example Corp, Principal Analyst, Senior Analyst, Analyst), jobs (from to), locations (Offices), and people (Martha, Richard, John). Roles are connected to companies, and companies are connected to jobs. Jobs are connected to people.

companies where Li worked?

Thank You!