

Lanzando una base de datos de grafos con Amazon Neptune

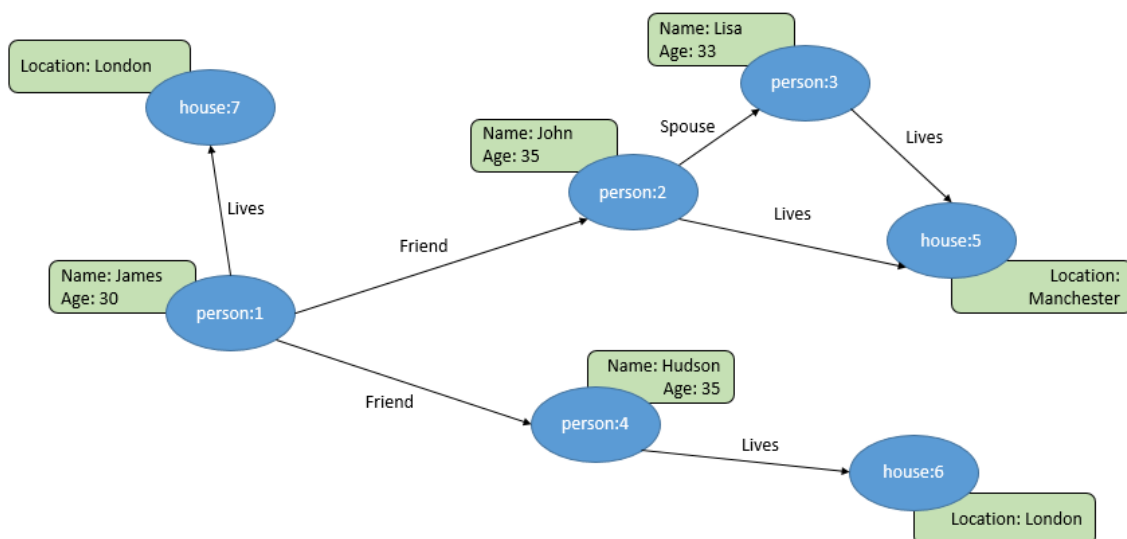
Amazon Neptune es un servicio de base de datos de gráficos administrados que crea y ejecuta aplicaciones que funcionan con conjuntos de datos altamente conectados. Es un motor de base de datos de gráficos de alto rendimiento diseñado específicamente para almacenar miles de millones de relaciones y consultar el gráfico con una latencia de milisegundos. La base de datos admite lenguajes de consulta como Apache TinkerPop Gremlin y SPARQL.

Crearé la siguiente estructura de datos de gráfico en este ejercicio. Tiene dos tipos de nodos: **persona** y **casa**

Estos nodos tienen tres relaciones o tipos de relaciones: **Vive con**, **Cónyuge** y **Amigo**

El nodo de tipo de **persona** tiene dos propiedades **Name** y **Age**

El nodo de tipo **Casa** tiene una propiedad **Ubicación**



Comienza con la creación de la base de datos de Amazon Neptune, que se utiliza para trabajar con los datos del gráfico.

Inicie sesión en la consola de AWS. Seleccione **Irlanda** como región.

Vaya a la consola de administración de Neptune y haga clic en el botón **Iniciar Amazon Neptune**.

Graph database

Create an Amazon Neptune database cluster

Launch Amazon Neptune

En la pantalla siguiente, mantenga la versión en la selección predeterminada. Escriba **nombre** para el identificador de clúster. Seleccione la opción **Desarrollo y prueba** para las plantillas. Mantenga el resto de la configuración por defecto y haga clic en el botón **Crear base de datos**

Engine type

neptune

Version [Info](#)

Neptune 1.0.4.1.R1 ▼

Settings

DB cluster identifier [Info](#)

Type a name for your DB cluster. The name must be unique cross all DB clusters owned by your AWS account in the current AWS Region.

dojographdb

The DB cluster identifier is case-insensitive, but is stored as all lowercase (as in "mydbcluster"). Constraints: 1 to 60 alphanumeric characters or hyphens. First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

Templates

Choose a template to meet your use case.

☐ **Production**

Use defaults for high availability and fast, consistent performance.

☒ **Development and Testing**

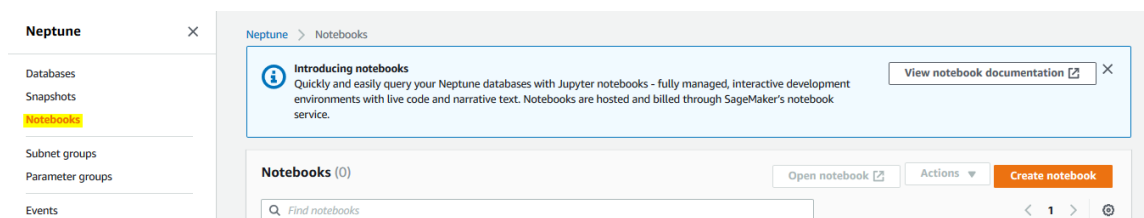
This instance is intended for development use outside of a production environment.

Comenzará la creación de la base de datos. Espere hasta que el estado de la base de datos cambie a **Disponible** .

Databases									
<div> <div>Group resources</div> <div>Modify</div> <div>Actions</div> <div>Create database</div> </div> <div>Filter databases</div>									
DB identifier	Role	Engine	Region & AZ	Size	Status	CPU	Current		
dojographdb	Cluster	Neptune	eu-west-1	-	Available	-			
dojographdb-instance-1	Writer	Neptune	eu-west-1a	db.t3.medium	Available	-			

La base de datos está lista. A continuación, creará un cuaderno con el lenguaje de consulta Gremlin.

En la consola de administración de Neptune, haga clic en la opción de menú **Cuadernos** a la izquierda y luego haga clic en el botón **Crear cuaderno**



En la pantalla siguiente, selecciona tu como clúster. Escribe **nombrenotebook** para el nombre del cuaderno. Selecciona la opción **Crear un rol de IAM**. Escribe **nombrenotebook** para el rol de IAM. Mantén el resto de la configuración por defecto y haz clic en el botón **Crear cuaderno**.

Notebook configuration

Cluster
Associate this notebook with an existing cluster. If no cluster exists, create a cluster

dojographdb

Create cluster

Notebook instance type

ml.t3.medium

Notebook name

aws-neptune-dojonotebook

Names may only contain letters (A-Z), numbers (0-9), or hyphens (-).

Description - optional

256 character max

IAM role name
Notebook instances require permissions to call other services including SageMaker and S3. Create a new role or update an existing role. You can also manage roles on the [IAM console](#)

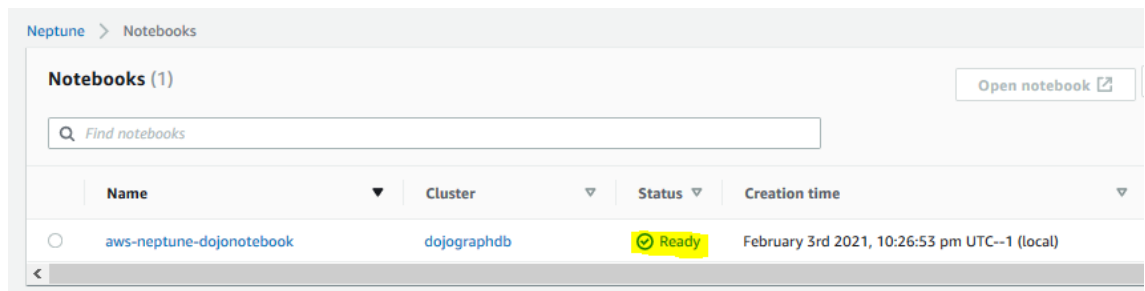
☒ Create an IAM role
 ☐ Choose an existing IAM role

IAM role

AWSNeptuneNotebookRole-dojonotebook

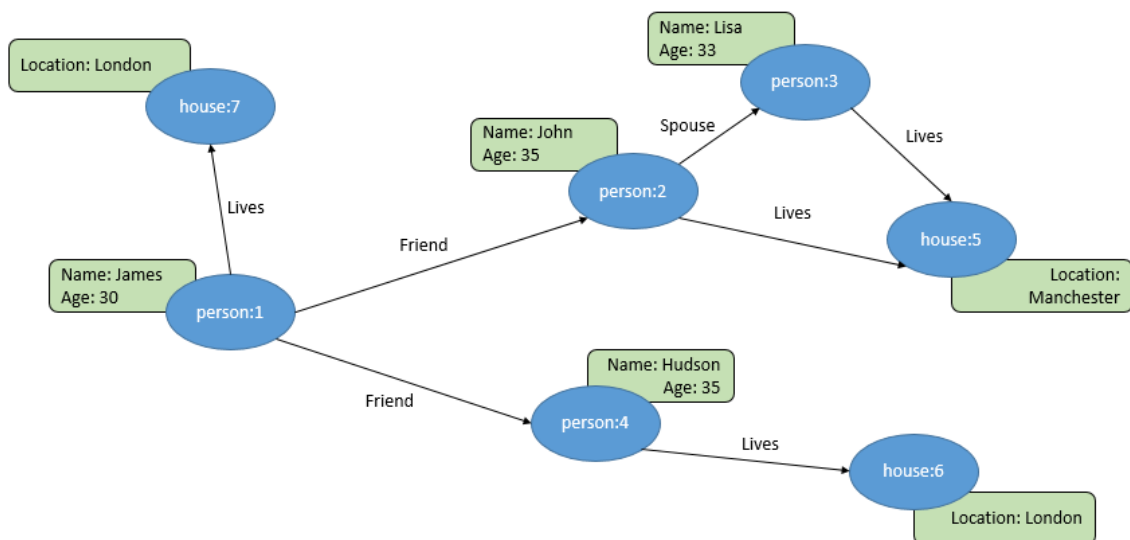
To create an IAM role you must have `CreateRole`, `CreatePolicy` and `AttachRolePolicy` permissions.

Comenzaría la creación del cuaderno. Espera hasta que el estado cambie a **Listo**.

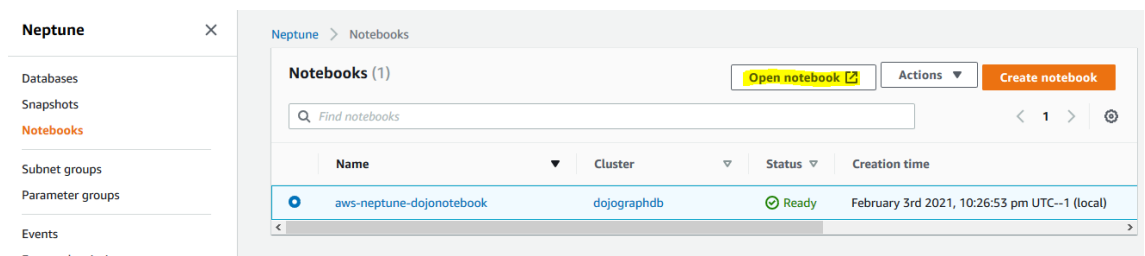


El cuaderno está listo. Utiliza el lenguaje de consulta Gremlin en el cuaderno para trabajar con los datos del grafo.

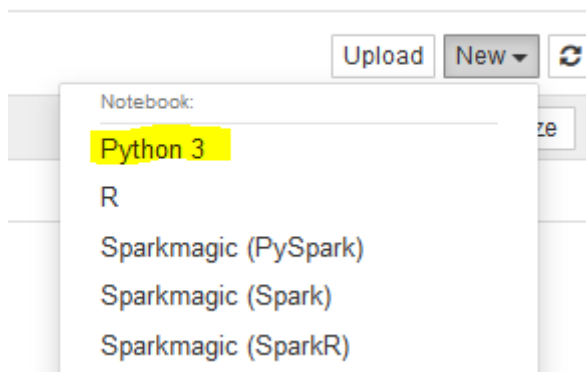
Los datos del grafo tienen el siguiente aspecto:



En la consola de administración de Neptune, haz clic en la opción de menú **Notebooks** a la izquierda. Selecciona el cuaderno y haz clic en el botón **Abrir cuaderno**.



Se abrirá Jupyter Notebook en una nueva ventana o pestaña del navegador. En Jupyter Notebook, haz clic en la opción **Python 3** en el menú **Nuevo**.



Abre el IDE del portátil en una nueva ventana o pestaña del navegador. Primero creará nodos de tipos de **personas** y **casas**.

Luego configura las relaciones (**Vive con**, **Amigo** y **Cónyuge**) entre ellos. Ejecuta el siguiente código para crear el nodo de tipo de primera persona junto con sus propiedades.



```
%%gremlin
g.addV('person').property(id, '1').property('Name', 'James').property('Age', '30')
```

Del mismo modo, ejecuta el siguiente código para crear el primer nodo de tipo de casa.

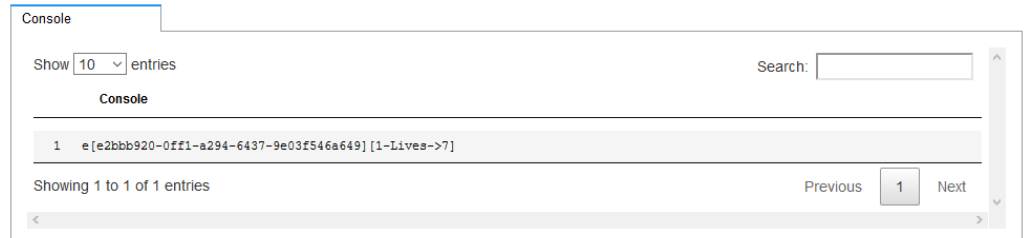


```
%%gremlin
g.addV('house').property(id, '7').property('Location', 'London')
```

Repite el paso anterior y ejecuta el código correspondiente para crear los nodos de persona y casa restantes.

Los nodos **V[1]** a **V[7]** están listos. Vamos a crear relaciones entre ellos. Ejecuta el siguiente código para crear una relación **de Vive con** entre el nodo **V[1]** y **V[7]** .

```
In [8]: %%gremlin
g.addE('Lives').from(g.V('1')).to(g.V('7'))
```



```
%%gremlin
g.addE('Lives').from(g.V('1')).to(g.V('7'))
```

De manera similar, ejecuta el siguiente código para crear una relación **de Amigo** entre el nodo **V[1]** y **V[2]** .

```
In [9]: %%gremlin
g.addE('Friend').from(g.V('1')).to(g.V('2'))
```

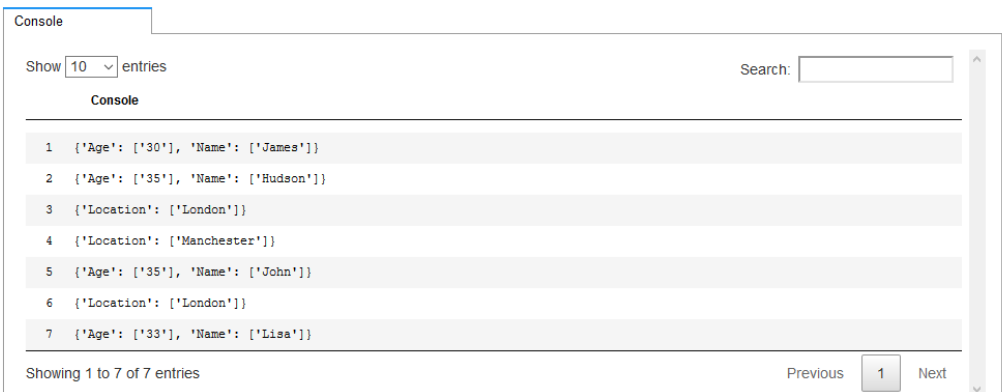


```
%%gremlin
g.addE('Friend').from(g.V('1')).to(g.V('2'))
```

Repite el paso anterior y ejecuta el código correspondiente para crear las relaciones restantes.

Los datos ya están creados. Ahora vamos a consultarlos. Comencemos enumerando todos los nodos en la base de datos de gráficos.

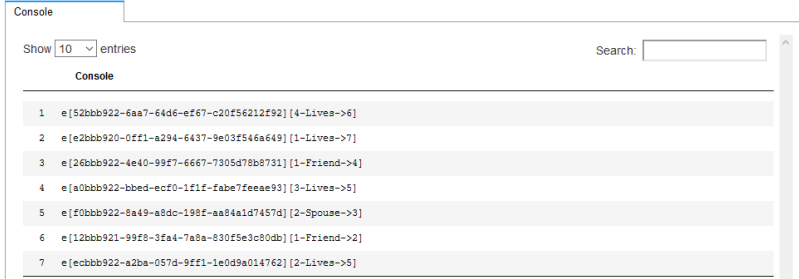
```
In [16]: %%gremlin
g.V().valueMap()
```



```
%%gremlin
g.V().valueMap()
```

Ejecuta el siguiente código para enumerar todos los bordes.

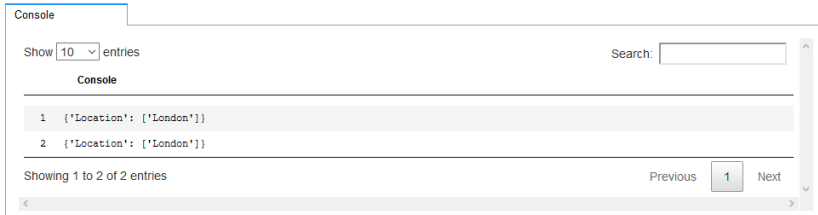
```
In [18]: %%gremlin
g.E()
```



```
%%gremlin
g.E()
```

Ejecuta la siguiente consulta para enumerar las **casas** donde **Location** es **London**.

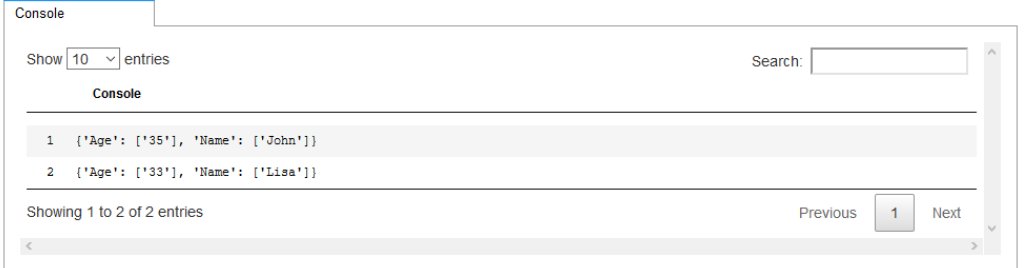
```
In [19]: %%gremlin
g.V().has('Location', 'London').valueMap()
```



```
%%gremlin
g.V().has('Location', 'London').valueMap()
```

Ejecuta la siguiente consulta para enumerar quién **vive** en la **casa** con **ubicación = Manchester** .

```
In [22]: %%gremlin
g.V().has('Location', 'Manchester').inE('Type','Lives').outV().valueMap()
```



	Console
1	{'Age': ['35'], 'Name': ['John']}
2	{'Age': ['33'], 'Name': ['Lisa']}

Showing 1 to 2 of 2 entries

Previous 1 Next

```
%%gremlin
g.V().has('Location', 'Manchester').inE('Type','Lives').outV().valueMap()
```

Ejecuta la siguiente consulta para enumerar la **casa** con su **ubicación** de las **personas** que tienen **edad = 35** .

```
1. %%gremlin
2. g.V().hasLabel('person').has('Age','35').outE('Type','Lives').inV().valueMap()
```

Estos fueron algunos ejemplos desde el punto de vista de la consulta. Ahora vamos a ver como actualizar los datos.

Una **persona** con el **nombre** de **Hudson** tiene una **edad** de **35** años. Ejecuta la siguiente consulta para actualizar la edad de la persona a **40**.

```
In [35]: %%gremlin
g.V().has('Name', 'Hudson').property(single,'Age','40')
```



	Console
1	v[4]

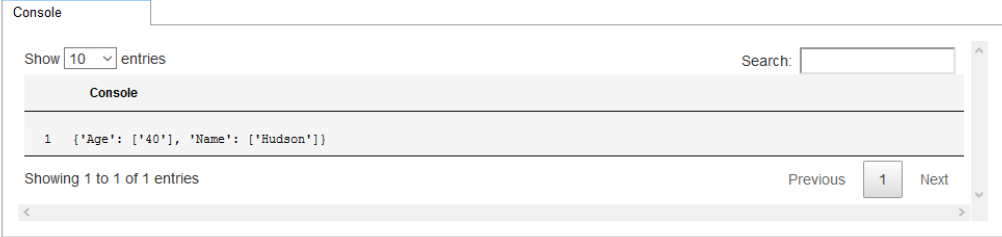
Showing 1 to 1 of 1 entries

Previous 1 Next

```
%%gremlin
g.V().has('Name', 'Hudson').property(single,'Age','40')
```


Puedes ejecutar el siguiente código para verificarlo.

```
In [36]: %%gremlin
g.V().has('Name', 'Hudson').valueMap()
```



The console output shows a table with one entry. The entry has a key 'Age' with value '40' and a key 'Name' with value 'Hudson'.

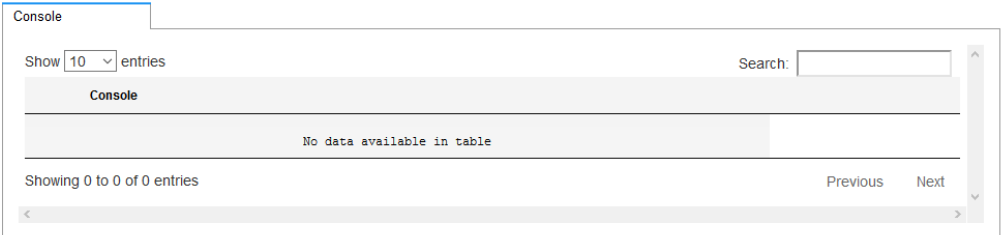
1	{'Age': ['40'], 'Name': ['Hudson']}

Showing 1 to 1 of 1 entries

```
%%gremlin
g.V().has('Name', 'Hudson').property(single, 'Age', '40')
```

Supongamos que deseamos eliminar a la **persona** cuyo **nombre** es **Hudson**. Ejecute la siguiente consulta para eliminar el nodo.

```
In [37]: %%gremlin
g.V().has('Name', 'Hudson').drop()
```



The console output shows a table with no entries. The message 'No data available in table' is displayed in the center of the table.

No data available in table	

Showing 0 to 0 of 0 entries

```
%%gremlin
g.V().has('Name', 'Hudson').drop()
```