Andres Bermeo

Como poder migrar desde una base de datos transaccional a BDOG (DE POSTGRES A neo4J)

Al derivar un modelo de gráfico a partir de un modelo relacional, debemos tener en cuenta un par de pautas
generales.

- Una fila es un nodo
- Un nombre de tabla es un nombre de etiqueta
- Una combinación o clave externa es una relación

¿En qué se diferencia el modelo gráfico del modelo relacional?
No hay nulos. Las entradas de valor no existentes (propiedades) simplemente no están presentes.

- Describe las relaciones con más detalle.
- Cualquiera de los modelos se puede normalizar más.
- Exportación de tablas relacionales a CSV

Una de las mas factibles es exportando nuestros datos de la base de datos relacional a archivos CSV.
Un formato común que muchos sistemas pueden manejar en un archivo plano de valores separados por comas
(CSV), así que veamos cómo exportar tablas relacionales desde una base de datos PostgreSQL a archivos
CSV para que podamos crear nuestro gráfico.
El comando 'copiar' de PostgreSQL nos permite ejecutar una consulta SQL y escribir el resultado en un archivo CSV. Podemos ensamblar un breve script .sql de estos comandos de copia, como se muestra a continuación.

```
COPY (SELECT * FROM customers) TO '/tmp/customers.csv' WITH CSV header; COPY (SELECT * FROM
suppliers) TO '/tmp/suppliers.csv' WITH CSV header; COPY (SELECT * FROM products) TO '/tmp/products.csv'
WITH CSV header; COPY (SELECT * FROM employees) TO '/tmp/employees.csv' WITH CSV header; COPY
(SELECT * FROM categories) TO '/tmp/categories.csv' WITH CSV header; COPY (SELECT * FROM orders LEFT
OUTER JOIN order_details ON order_details.OrderID = orders.OrderID) TO '/tmp/orders.csv' WITH CSV header;
```

Importando los datos usando Cypher

Después de haber exportado nuestros datos desde PostgreSQL, usaremos el comando LOAD CSV de Cypher
para transformar el contenido del archivo CSV en una estructura de gráfico. Primero, probablemente querremos
colocar nuestros archivos CSV en un directorio de fácil acceso. Con Neo4j Desktop, podemos colocarlos en el
directorio de importación de la base de datos local (instrucciones detalladas que se encuentran en nuestra guía
de importación de escritorio). De esta manera, podemos usar el file:///prefijo en nuestras declaraciones Cypher
para ubicar los archivos. También podemos colocar los archivos en otro directorio local o remoto (admite
HTTPS, HTTP y FTP) y especificar la ruta completa en nuestras declaraciones Cypher. Dado que estamos
usando Neo4j Desktop en este ejemplo, usaremos la carpeta de importación para la base de datos y la ruta de
nuestros archivos CSV puede comenzar con el file:///prefijo.
Ahora que tenemos nuestros archivos a los que podemos acceder fácilmente, podemos usar el LOAD
CSVcomando de Cypher para leer cada archivo y agregar declaraciones Cypher después para tomar los datos de fila / columna y transformarlos en el gráfico.
El script Cypher completo está disponible en Github para que lo copie y ejecute, pero repasaremos cada sección a continuación para explicar qué hace cada pieza del script.

```
// Create orders LOAD CSV WITH HEADERS FROM 'file:///orders.csv' AS row
MERGE (order:Order {orderID: row.OrderID}) ON CREATE SET order.shipName =
row.ShipName; // Create products LOAD CSV WITH HEADERS FROM
'file:///products.csv' AS row MERGE (product:Product {productID: row.ProductID})
ON CREATE SET product.productName = row.ProductName, product.unitPrice =
toFloat(row.UnitPrice); // Create suppliers LOAD CSV WITH HEADERS FROM
'file:///suppliers.csv' AS row MERGE (supplier:Supplier {supplierID: row.SupplierID})
ON CREATE SET supplier.companyName = row.CompanyName; // Create employees
LOAD CSV WITH HEADERS FROM 'file:///employees.csv' AS row MERGE
(e:Employee {employeeID:row.EmployeeID}) ON CREATE SET e.firstName =
row.FirstName, e.lastName = row.LastName, e.title = row.Title; // Create categories
LOAD CSV WITH HEADERS FROM 'file:///categories.csv' AS row MERGE
(c:Category {categoryID: row.CategoryID}) ON CREATE SET c.categoryName =
row.CategoryName, c.description = row.Description;
```