

Intercambio de datos de Bases de datos relacionales a grafos de propiedad

¹Manuel Rios

¹Facultad de Ciencias,
Universidad Nacional de Ingeniería

01 Agosto 2023

Importancia de las bases de datos orientado a grafos de propiedad



Figura: Empresas que utilizan BD orientado a grafos de propiedad

Análisis de datos en BD orientado a grafos de propiedad



Figura: Análisis de datos

Áreas de estudio

- ▶ Customer Journey.
- ▶ Knowledge Graph.
- ▶ Mapas estratégicos de riesgo.
- ▶ Lucha contra el Crimen Financiero PBC – AML

¿Objetivos?

- ▶ Comprensión del esquema y restricciones en una BD relacional.
- ▶ Comprensión del esquema y restricciones de un grafo con propiedad.
- ▶ Formalización de la BD relacional.
- ▶ Formalización del grafo con propiedad.
- ▶ Formalización del mapeo entre BD relacional a Grafo con propiedad.

Base de Datos Relacionales

ID	JOBTITLE	EMAILADDRESS	FIRSTNAMELASTNAME
5869	Ambulatory Nurse	Marilyn_Clarke7275@sveldo.biz	Marilyn Clarke
5870	Call Center Representative	Drew_Ianson5964@twipet.com	Drew Ianson
5871	Executive Director	Maxwell_Vallins6716@sveldo.biz	Maxwell Vallins
5872	Doctor	Kieth_Gordon1073@bulaffy.com	Kieth Gordon
5873	Budget Analyst	Matthew_Rehman3766@twipet.com	Matthew Rehman
5874	Audiologist	David_Weldon1163@tonsy.org	David Weldon

Figura: Tabla de una base de datos relacional

Lenguaje SQL

```
141  
142  
143 CREATE TABLE IF NOT EXISTS `wp_ngg_pictures` (  
144   `pid` bigint(20) NOT NULL AUTO_INCREMENT,  
145   `image_slug` varchar(255) NOT NULL,  
146   `post_id` bigint(20) NOT NULL DEFAULT '0',  
147   `galleryid` bigint(20) NOT NULL DEFAULT '0',  
148   `filename` varchar(255) NOT NULL,  
149   `description` mediumtext,  
150   `alttext` mediumtext,  
151   `imagedate` datetime NOT NULL DEFAULT '0000-00-00'  
152   `exclude` tinyint(4) DEFAULT '0',  
153   `sortorder` bigint(20) NOT NULL DEFAULT '0'
```

Figura: Lenguaje de base de datos relacionales, SQL

Base de Datos Orientado a grafos de propiedad

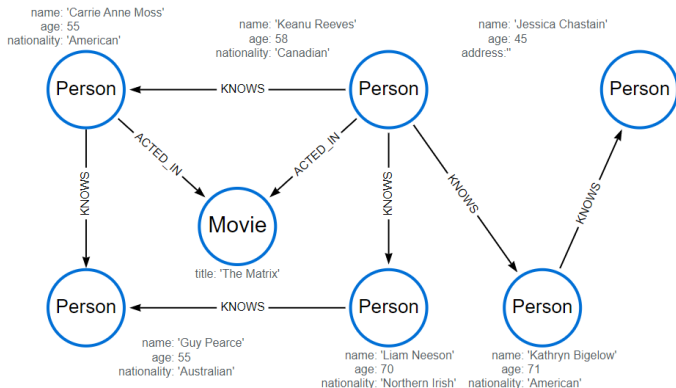


Figura: Base de datos orientado a grafos de propiedad

Lenguaje Cypher

Neo4j Cypher Query Language

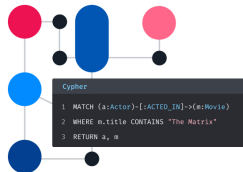


Figura: Cypher Query Language, lenguaje de consulta para Neo4j

Lenguaje Cypher

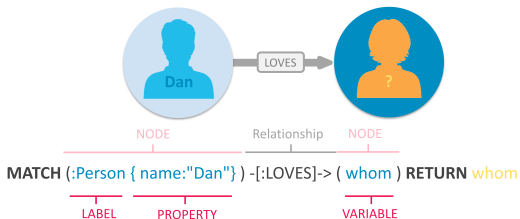


Figura: Ejemplo de lenguaje Cypher

Lenguaje de Programación Python



Figura: Python

Estudio de la estructura de la base de datos relacional

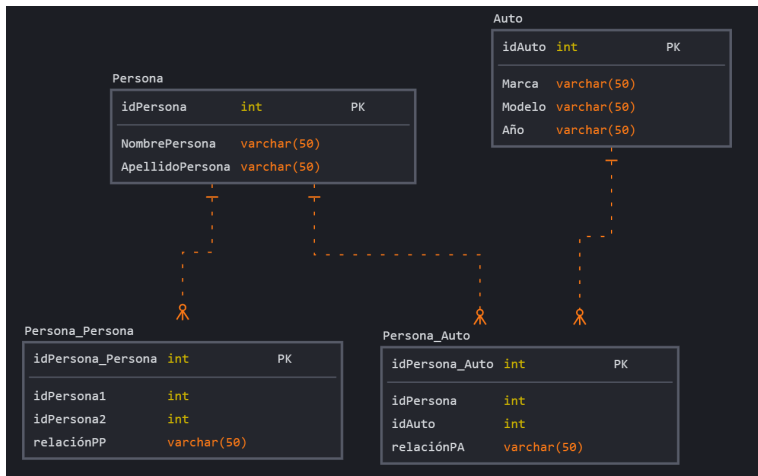


Figura: Diagrama de la base de datos

Restricciones

- ▶ Se tendrá 2 tipos de tablas para las entidades, el primer tipo será las entidades que no tengan dependencias y el segundo tipo serán las entidades que dependan de otras.
- ▶ El segundo tipo de tablas donde las entidades dependan de otras serán las tablas que se caracterizan por tener las relaciones de otras entidades y el nombre de éstas tendrán la siguiente estructura: Entidad1_Entidad2 .
- ▶ Este segundo tipo de tabla definido en el anterior punto tendrá la siguiente estructura idTabla, idEntidad1, idEntidad2, relación.

Estudio de la estructura de la base de datos orientado a grafos de propiedad

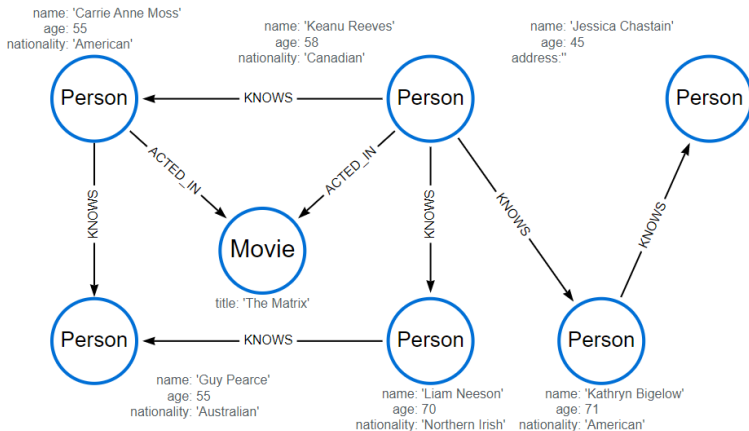


Figura: Estructura de la base de datos orientada a grafos de propiedad

Formalismo lógico

Para la tabla Persona tenemos el siguiente formalismo lógico:

$$\text{Persona}(\text{idPersona}, \text{NombrePersona}, \text{ApellidoPersona}) \Rightarrow \text{Tripla}(\text{idPersona}, \text{NombrePersona}, \text{valor}(\text{NombrePersona})) \wedge \text{Tripla}(\text{idPersona}, \text{ApellidoPersona}, \text{valor}(\text{ApellidoPersona}))$$

Para la tabla Auto tenemos el siguiente formalismo lógico:

$$\text{Auto}(\text{idAuto}, \text{Marca}, \text{Modelo}, \text{Año}) \Rightarrow \text{Tripla}(\text{idAuto}, \text{Marca}, \text{valor}(\text{Marca})) \wedge \text{Tripla}(\text{idAuto}, \text{Modelo}, \text{valor}(\text{Modelo})) \wedge \text{Tripla}(\text{idAuto}, \text{Año}, \text{valor}(\text{Año}))$$

Figura: Formalismo lógico de las tablas Persona y Auto

Mapeo Directo (Direct Mapping)




Figura: Registros de las tablas que serán nodos

Mapeo Directo (Direct Mapping)

Results		Messages	
	idPersona	nombrePersona	apellidoPersona
1	1	Noah	Taylor
2	2	Maya	Morris
3	3	Bella	Gonzalez
4	4	Grace	Thompson
5	5	Sofia	Wood
6	6	Oliver	Clark
7	7	Jack	Rogers
8	8	Savannah	Reed
9	9	Mia	Brown
10	10	Eli	Lopez
11	11	Luke	Phillips

Figura: Registros de las tablas que serán nodos

Mapeo Directo (Direct Mapping)



The screenshot displays a graph node interface. On the left, there is a light blue square containing an orange circle with the text "Martinez". To the right of this square is a white panel titled "Node properties" with a right-pointing arrow. Inside this panel, there is an orange pill-shaped button labeled "Persona". Below the button is a table of properties for the node.



<id>	88	
apellidoPersona	Collins	
idPersona	89	
nombrePersona	Riley	

Figura: Nodo de tipo Persona

Mapeo Directo (Direct Mapping)

Persona_Persona		
idPersona_Persona	int	PK
idPersona1	int	
idPersona2	int	
relaciónPP	varchar(50)	

Persona_Auto		
idPersona_Auto	int	PK
idPersona	int	
idAuto	int	
relaciónPA	varchar(50)	

Figura: Registros de las tablas que serán relaciones

Mapeo Directo (Direct Mapping)



<div><div> Results</div><div> Messages</div></div>				
	idPersona_Persona	idPersona1	idPersona2	relacionPP
1	1	1	204	Conoce
2	2	2	661	Pareja_de
3	3	3	188	Director_de
4	4	4	586	Jefe_de
5	5	5	451	Estudiante_de
6	6	6	496	Cocinero_de
7	7	7	473	Amigo_de
8	8	8	109	Manager_de
9	9	9	52	Vecino_de
10	10	10	100	Amante_de

Figura: Registros de las tablas que serán relaciones

Mapeo Directo (Direct Mapping)

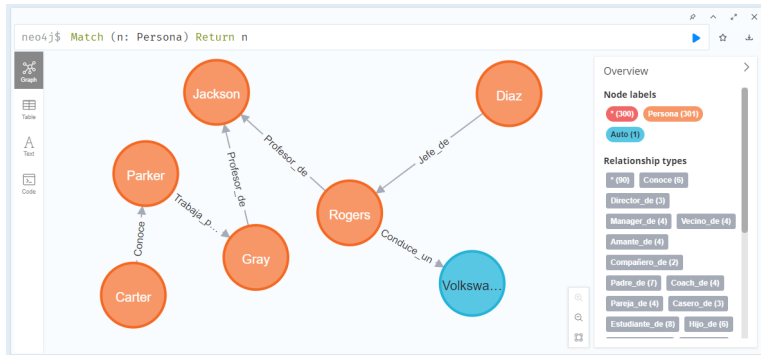
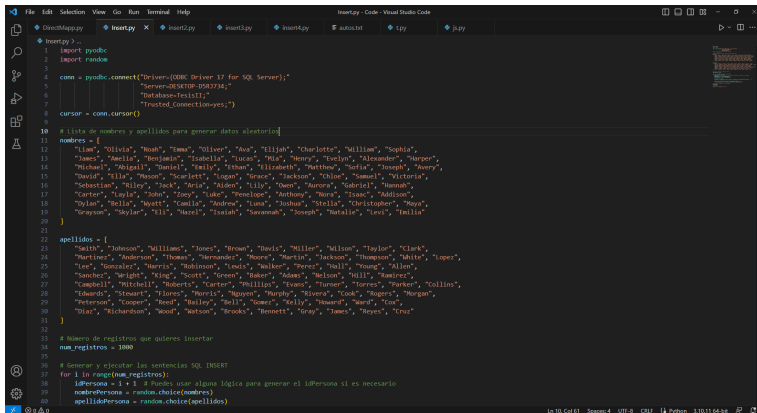


Figura: Nodos Persona y Auto con sus relaciones

Creación de la base de datos relacional



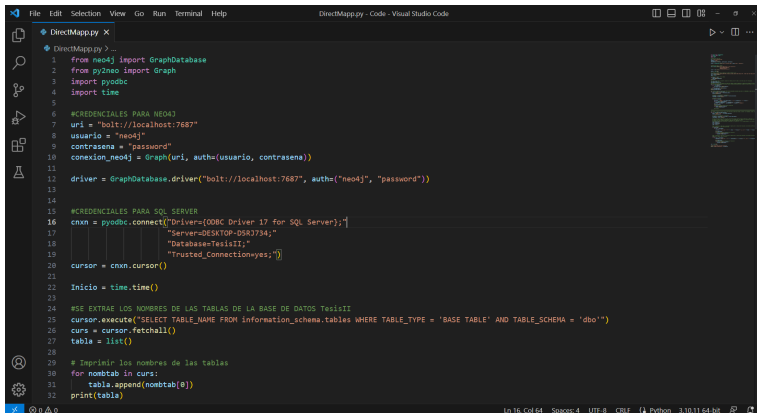
```
File Edit Selection View Go Run Terminal Help
Insertpy - Code - Visual Studio Code

DirectMapping Insertpy X insert2.py insert3.py insert4.py autos.txt tipy jupy

Insertpy > ...
1 import pyodbc
2 import random
3
4 conn = pyodbc.connect("Driver={ODBC Driver 17 for SQL Server};"
5                      "Server=DESKTOP-D587734;"
6                      "Database=TestII;"
7                      "Trusted_Connection=yes;")
8 cursor = conn.cursor()
9
10 # lista de nombres y apellidos para generar datos aleatorios
11 nombres = [
12     "Lise", "Olivia", "Ruth", "Emme", "Oliver", "Ava", "Elijah", "Charlotte", "William", "Sophia",
13     "James", "Amelia", "Benjamin", "Isabella", "Lucas", "Mia", "Henry", "Evelyn", "Alexander", "Harper",
14     "Michael", "Abigail", "Daniel", "Emily", "Ethan", "Elizabeth", "Matthew", "Sofia", "Joseph", "Avery",
15     "David", "Ellie", "Mason", "Scarlett", "Logan", "Grace", "Jackson", "Chloe", "Samuel", "Victoria",
16     "Sebastian", "Riley", "Jack", "Aria", "Aiden", "Lily", "Owen", "Aurora", "Gabriel", "Hannah",
17     "Carter", "Lyla", "Sofia", "Zoe", "Luke", "Penelope", "Anthony", "Mia", "Isaac", "Addison",
18     "Dylan", "Melia", "Natalie", "Camila", "Andreea", "Liam", "Joshua", "Stella", "Christopher", "Maya",
19     "Grayson", "Skylar", "Eli", "Mazel", "Isaiah", "Savannah", "Joseph", "Natalie", "Levi", "Isella"
20 ]
21
22 apellidos = [
23     "Smith", "Johnson", "Williams", "Jones", "Brown", "Davis", "Miller", "Wilson", "Taylor", "Clark",
24     "Martinez", "Anderson", "Thomas", "Hernandez", "Moore", "Martin", "Jackson", "Thompson", "White", "Lopez",
25     "Lee", "Gonzalez", "Harris", "Robinson", "Lewis", "Walker", "Perez", "Hall", "Young", "Allen",
26     "Sanchez", "Wright", "King", "Scott", "Green", "Baker", "Adams", "Nelson", "Hill", "Ramirez",
27     "Cowell", "Mitchell", "Roberts", "Carter", "Phillips", "Evans", "Turner", "Torres", "Parker", "Collins",
28     "Edwards", "Stewart", "Flores", "Harris", "Nguyen", "Murphy", "Rivera", "Cook", "Burgers", "Morgan",
29     "Peterson", "Cooper", "Reed", "Bailey", "Bell", "Gomez", "Kelly", "Howard", "Ward", "Cox",
30     "Diaz", "Richardson", "Wood", "Watson", "Brooks", "Bennett", "Gray", "James", "Reyes", "Cruz"
31 ]
32
33 # Numero de registros que quieres insertar
34 num_registros = 1000
35
36 # Generar y ejecutar las sentencias SQL INSERT
37 for i in range(num_registros):
38     idPersona = i + 1 # Puedes usar alguna logica para generar el idPersona si es necesario
39     nombrePersona = random.choice(nombres)
40     apellidoPersona = random.choice(apellidos)
```

Figura: Creación de base de datos por medio de Scripts

Creación del código para el intercambio de datos



```
1 from neo4j import GraphDatabase
2 from py2neo import Graph
3 import pyodbc
4 import time
5
6 #CREDENCIALES PARA NEO4J
7 uri = "bolt://localhost:7687"
8 usuario = "neo4j"
9 contrasena = "password"
10 conexion_neo4j = Graph(uri, auth=(usuario, contrasena))
11
12 driver = GraphDatabase.driver("bolt://localhost:7687", auth=("neo4j", "password"))
13
14
15 #CREDENCIALES PARA SQL SERVER
16 cnxn = pyodbc.connect("Driver={ODBC Driver 17 for SQL Server};"
17                       "Server=DESKTOP-D5R3734;"
18                       "Database=TesisII;"
19                       "Trusted_Connection=yes;"}
20 cursor = cnxn.cursor()
21
22 Inicio = time.time()
23
24 #SE EXTRAE LOS NOMBRES DE LAS TABLAS DE LA BASE DE DATOS TesisII
25 cursor.execute("SELECT TABLE_NAME FROM information_schema.tables WHERE TABLE_TYPE = 'BASE TABLE' AND TABLE_SCHEMA = 'dbo'")
26 curs = cursor.fetchall()
27 tabla = list()
28
29 # Imprimir los nombres de las tablas
30 for nombtab in curs:
31     tabla.append(nombtab[0])
32 print(tabla)
```

Ln 16, Col 64 Spaces: 4 UTF-8 CRLF Python 3.10.11 64-bit

Figura: Código de intercambio de datos

Resultados

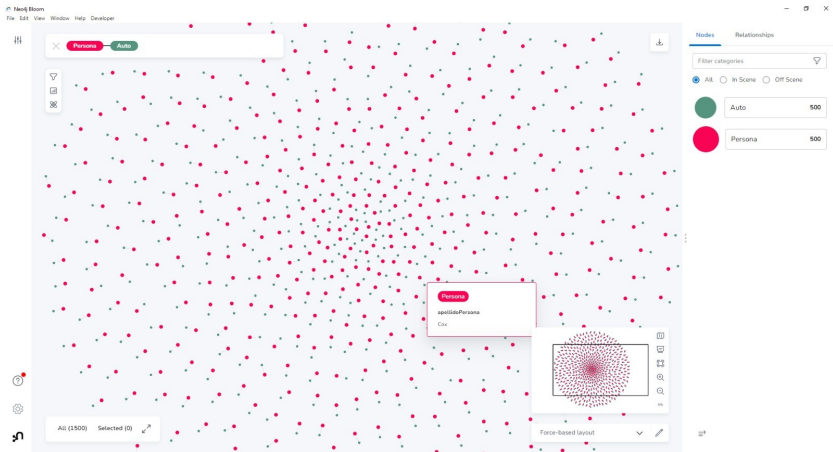


Figura: Nodos Persona y Auto con su respectiva relación creada en Neo4j

Resultados

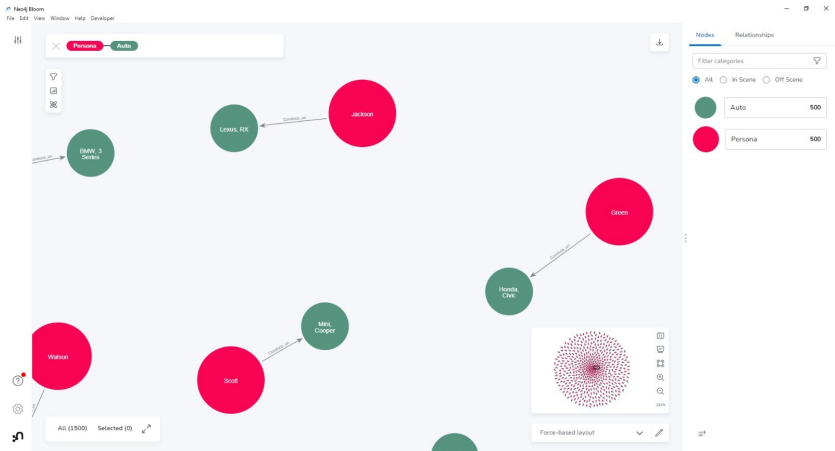


Figura: Nodos Persona y Auto con su respectiva relación creada en Neo4j

Resultados

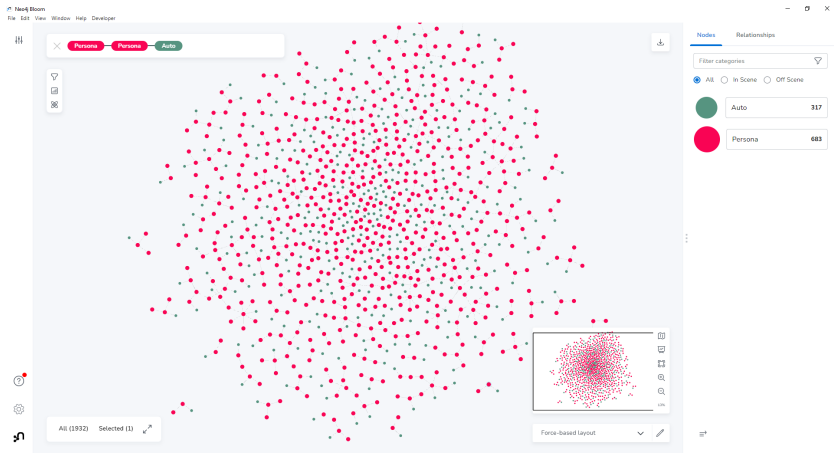


Figura: Grafo Persona-Persona-Auto en Neo4j

Resultados

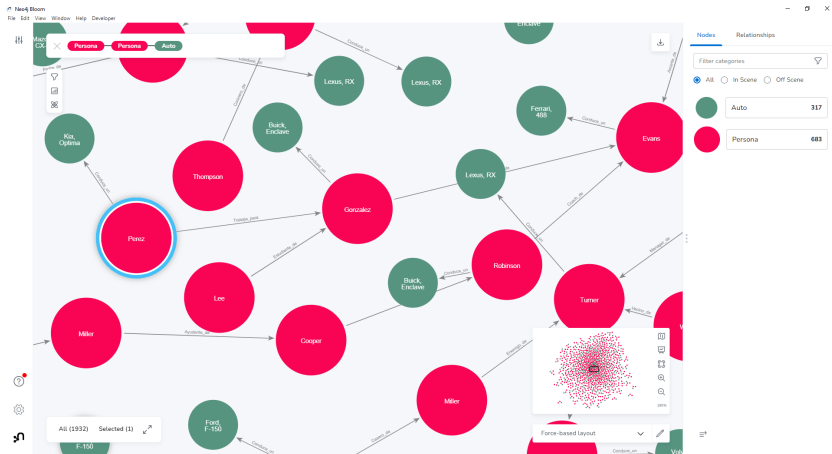


Figura: Grafo Persona-Persona-Auto en Neo4j

Resultados



Figura: Vista de algunas relaciones creadas y cantidad

Resultados

```
Match (n1 { idPersona:997}), (n2 {idPersona:454}) CREATE (n1)-[:Hijo_de]->(n2)
Match (n1 { idPersona:998}), (n2 {idPersona:13}) CREATE (n1)-[:Casero_de]->(n2)
Match (n1 { idPersona:999}), (n2 {idPersona:933}) CREATE (n1)-[:Amante_de]->(n2)
Match (n1 { idPersona:1000}), (n2 {idPersona:37}) CREATE (n1)-[:Jefe_de]->(n2)
```

Intercambio de datos exitosa

Tiempo de ejecución: 158.33262276649475
PS C:\Users\Skyline\Desktop\UNI\Tesis II\Code>

Figura: Tiempo de ejecución del script de intercambio de datos

conclusiones

- ▶ Con la realización de este trabajo de investigación para la ejecución del mapeo directo se concluye que es necesario establecer una serie de pasos que conlleva a realizar el intercambio de datos.
- ▶ El intercambio de datos que se llevó a cabo depende de la estructura de la base de datos relacional.
- ▶ El esquema de la base de datos orientada a grafos de propiedad no tienen restricciones en cuanto a su estructura a la hora de realizar el intercambio de datos, por lo que a la hora de la creación del código se utilizan queries Cypher que no son complejos.
- ▶ Se obtuvo con éxito el intercambio de datos.

Trabajos futuros

- ▶ Intercambio de dato utilizando R2RML (RDB to RDF mapping language) un lenguaje estandarizado por W3C, donde RDF significa Resource Description Framework una estructura de datos estándar.
- ▶ Análisis de datos en una base de datos orientada a grafos de propiedad en Neo4j Bloom que tiene como origen una base de datos relacional.