

Paweł Rajba

pawel@cs.uni.wroc.pl

<http://pawel.ii.uni.wroc.pl/>

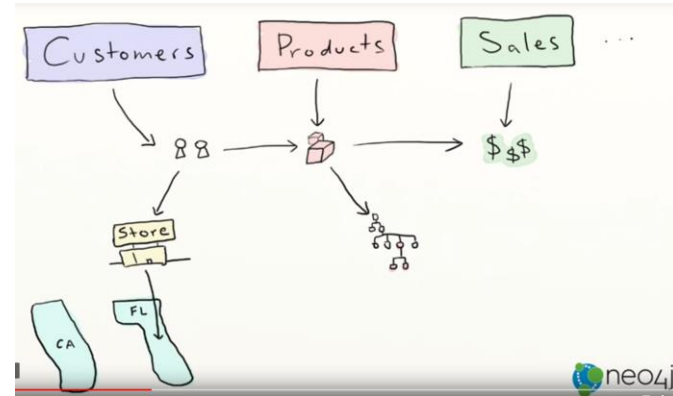
Neo4j

Agenda

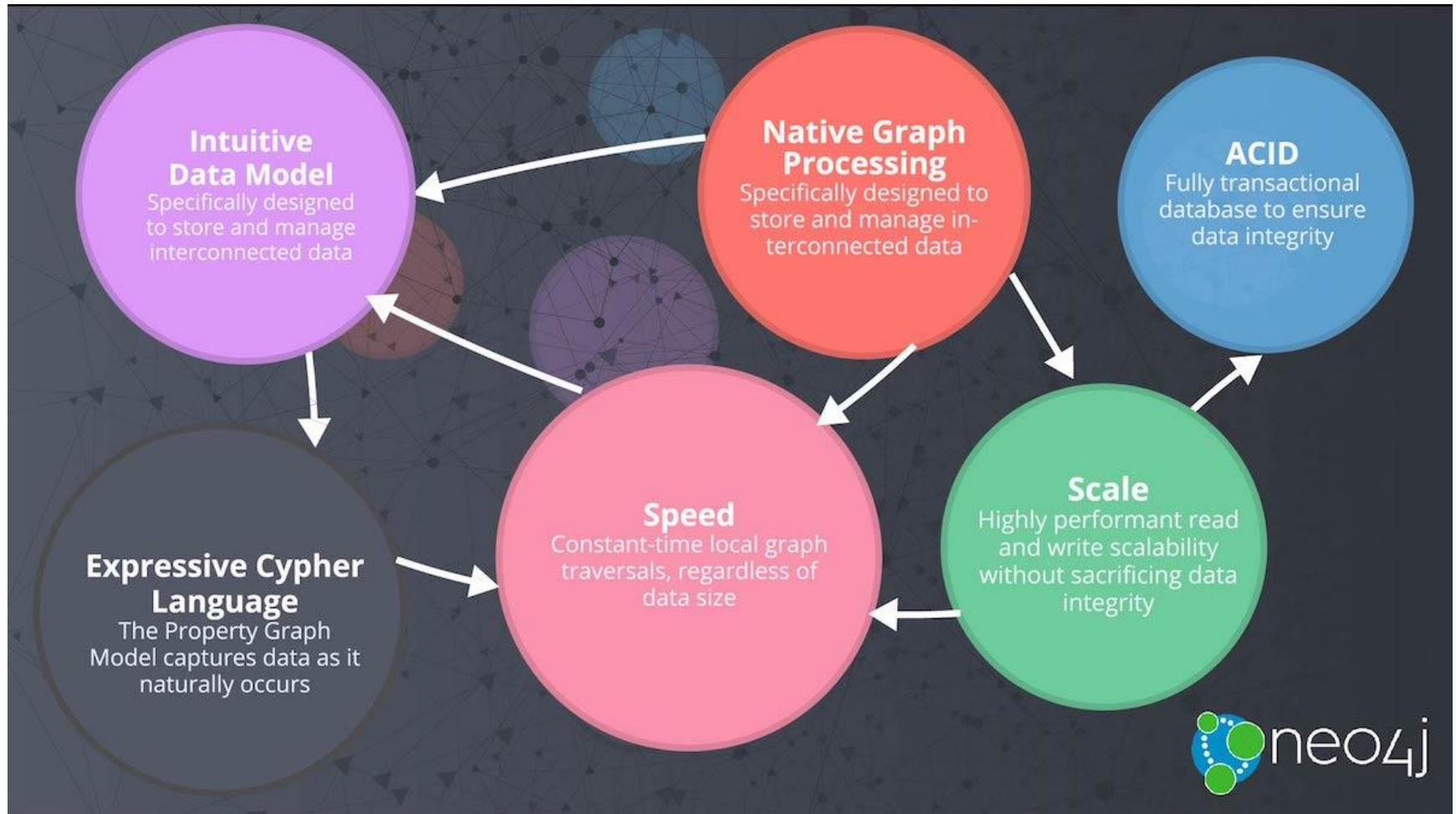
- Introduction
- Graph model
- Properties of graph database
- Cypher and example database

Introduction

- Website: <https://neo4j.com/>
- A graph database
 - Focusing primarily on relations
- Cypher as a query language
 - With roots in SQL
- Drivers for Popular Programming Languages
 - .Net, Java (also Spring), JavaScript, and Python
 - Communication based on binary “Bolt” protocol







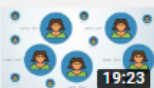

Introduction



Introduction

- A series of introduction videos available

<https://www.youtube.com/playlist?list=PL9Hl4pk2FsvWM9GWaguRhICQ-pa-ERd4U>

1		Intro to Graph Databases Episode #1 - Evolution of DBs Neo4j 5:12
2		Intro to Graph Databases Episode #2 - Properties of Graph DBs & Use Cases Neo4j 13:59
3		Intro to Graph Databases Episode #3 - Property Graph Model Neo4j 8:33
4		Intro to Graph Databases Episode #4 - (RDBMS+SQL) to (Graphs+Cypher) Neo4j 17:06
5		Intro to Graph Databases Episode #5 - Cypher, the Graph Query Language Neo4j 19:23
6		Intro to Graph Databases Episode #6 - Continuing with Cypher Neo4j 14:24

Introduction

NEO4j USE CASES

Real Time Recommendations

Master Data Management

Fraud Detection

Graph Based Search





















Network & IT-Operations

Identity & Access Management

Filter by Industry: All Industries

Filter by Use Case: All Use Cases

Search by Name: Company name

 Walmart	 Comcast	 Adobe	 eBay
 The ICIJ – Panama Papers	 Digitate	 Lyft	 Lockheed Martin Space
 U.S. Army	 Dun & Bradstreet	 Global 50 Bank	 German Centre for Diabetes Research
 UBS	 Perform Media	 Airbnb	 Microsoft
 MONSANTO	 IBM	 ICIJ	 novo nordisk

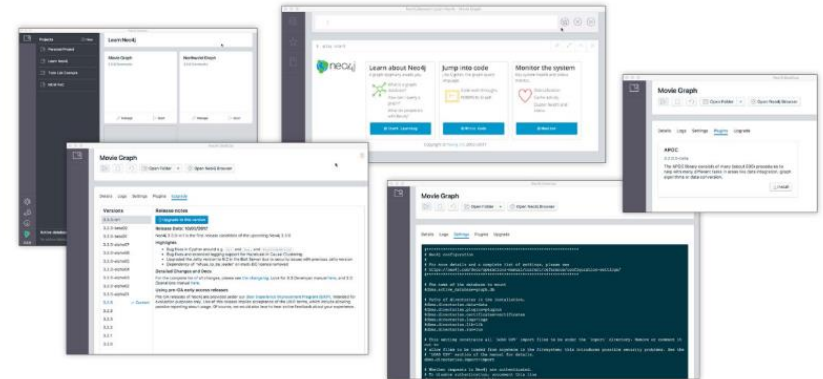
Introduction

■ Releases

Server Architecture	Community	Enterprise	Neo4j Aura
Native graph processing & storage	✓	✓	✓
Graph size limitations	34B nodes	None	34B nodes
ACID Transactions	✓	✓	✓
Node key schema constraints	-	✓	✓
Property existence constraints	-	✓	✓
Auto Reuse of Deleted Space*	✓	✓	-
Transactional ID Management*	✓	✓	-
Transaction Memory Constraint*	-	✓	-
Labeled property graph model	✓	✓	✓
System database*	✓	✓	-
High Performance Caching	✓	✓	✓
Cypher Graph Query Language	✓	✓	✓
Parallel Cypher Runtime*	-	✓	-
Cost-based Query Optimizer	✓	✓	✓
Cypher Sub-queries*	✓	✓	-
Cypher Query Tracing	-	✓	✓

*New in Version 4.0

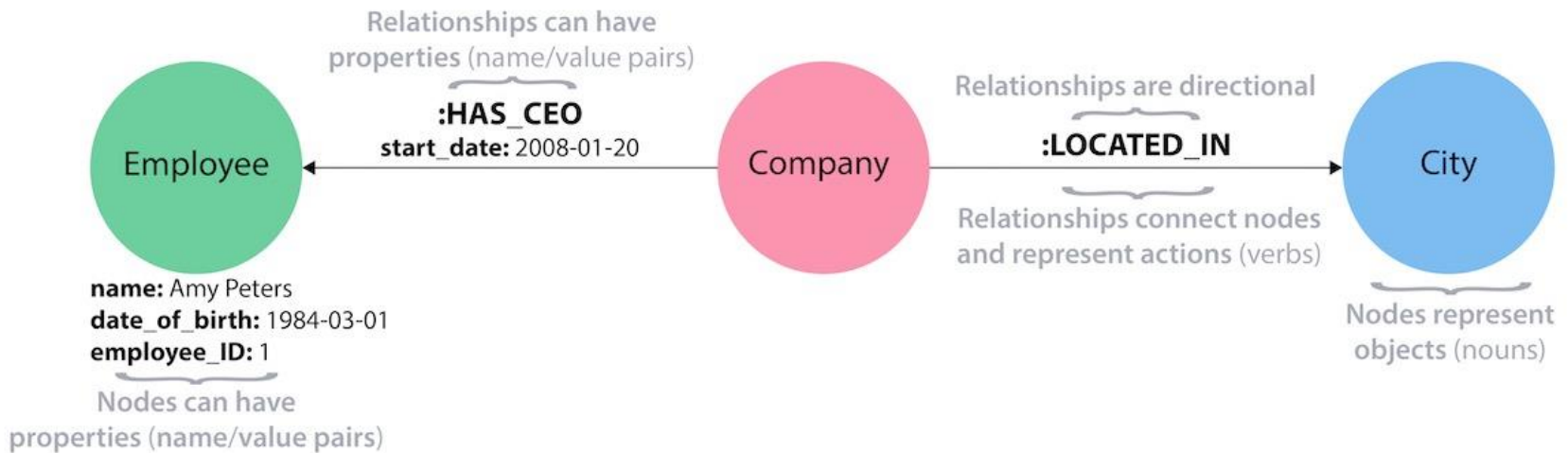
Neo4j Desktop: Developer-Friendly Packaging



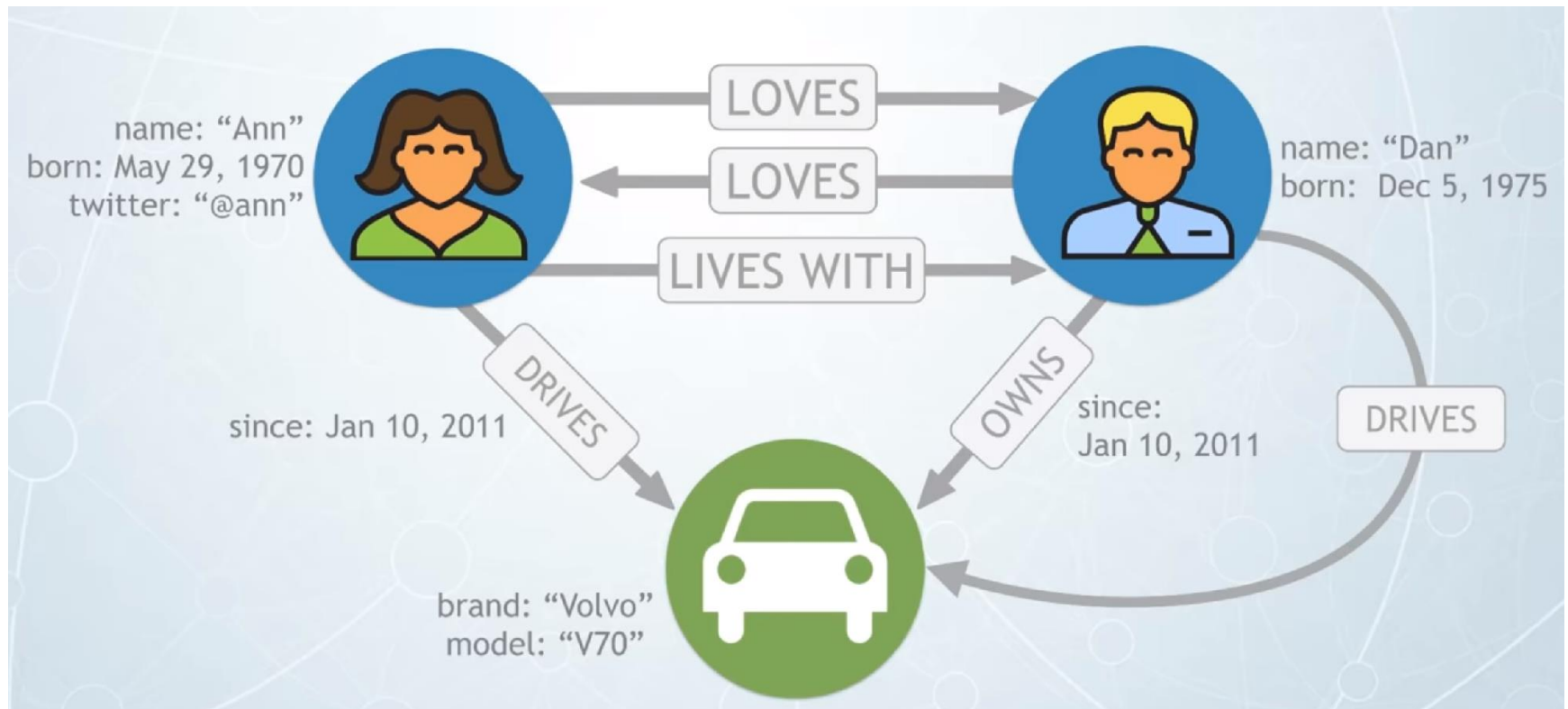
Graph model

- Treat the relationships between data as equally important to the data itself
- Rel. databases compute relationships at query time through JOIN operations
 - A graph database stores connections alongside the data in the model
- Key constituents of the graph model
 - Nodes – entities in the graph
 - Relationships – directed, named connections between two nodes (e.g. Employee *WORKS_FOR* Company)
 - Even if directed, they can be navigated in both directions
 - Properties – key/value pairs that can be attached both to nodes and relationships
 - Labels on nodes – representing types of nodes

Graph model



Graph model



Properties of graph databases

- Criterias to evaluate
 - Intuitiveness
 - Easier to take a journey from requirements and whiteboard to the actual data model
 - graph is natural way of expressing thoughts and though less translations are needed
 - Speed
 - Simpler model, so much quicker from the idea to the deployment
 - Ebay: Neo4j 1000x faster than MySQL based solution with 10-100 less code
 - Agility
 - No schemas, a naturally adaptive model (if sth needed, just add it)
 - Cypher language as a more concise way to express queries
 - What enables quicker understanding and easier way to change

Properties of graph databases

■ Graph & Cypher power*

Typical Complex SQL Join

```
[SELECT T.directReportees AS directReportees, sum(T.count) AS count
FROM (
  SELECT manager.pid AS directReportees, 0 AS count
  FROM person_reportee manager
   WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
  UNION
   SELECT manager.pid AS directReportees, count(manager.directly_manages) AS count
  FROM person_reportee manager
   WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
  GROUP BY directReportees
  UNION
   SELECT manager.pid AS directReportees, count(reportee.directly_manages) AS count
  FROM person_reportee manager
   JOIN person_reportee reportee
   ON manager.directly_manages = reportee.pid
   WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
  GROUP BY directReportees
  UNION
   SELECT manager.pid AS directReportees, count(l2Reportees.directly_manages) AS count
  FROM person_reportee manager
   JOIN person_reportee l1Reportees
   ON manager.directly_manages = l1Reportees.pid
   JOIN person_reportee l2Reportees
   ON l1Reportees.directly_manages = l2Reportees.pid
   WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
  GROUP BY directReportees
  ) AS T
 GROUP BY directReportees)
UNION
(SELECT T.directReportees AS directReportees, sum(T.count) AS count
FROM (
  SELECT manager.directly_manages AS directReportees, 0 AS count
  FROM person_reportee manager
   WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
  UNION
   SELECT reportee.pid AS directReportees, count(reportee.directly_manages) AS count
  FROM person_reportee manager
   JOIN person_reportee reportee
   ON manager.directly_manages = reportee.pid
   WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
  GROUP BY directReportees
  UNION
   SELECT l2Reportees.directly_manages AS directReportees, 0 AS count
  FROM person_reportee manager
   JOIN person_reportee l1Reportees
   ON manager.directly_manages = l1Reportees.pid
   JOIN person_reportee l2Reportees
   ON l1Reportees.directly_manages = l2Reportees.pid
   WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
  GROUP BY directReportees
  ) AS T
 GROUP BY directReportees)
```

The Same Query using Cypher

```
MATCH (boss)-[:MANAGES*0..3]->(sub),
      (sub)-[:MANAGES*1..3]->(report)
WHERE boss.name = "John Doe"
RETURN sub.name AS Subordinate,
       count(report) AS Total
```

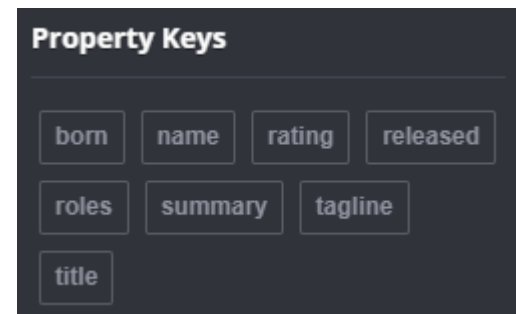
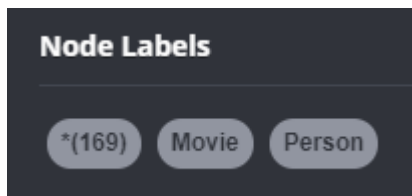
*Assuming the SQL query is written in the optimal way what probably is not the case

Example database

- Movies database
 - <https://neo4j.com/developer/example-project/>
- Domain model

```
(:Person {name})-[:ACTED_IN {roles}]->(:Movie {title,released})
```

- What's inside?



Cypher

- Open language
 - <http://www.opencypher.org/>
- Useful command
 - :help CREATE, :help MATCH, :help STH
- Based on ASCII Art and patterns
- Basic pattern
 - `()-[:RELATIONSHIP]->()`
 - Node – REL → Node
- Query
 - MATCH pattern
 - WHERE conditions
 - RETURN result

Cypher

- AsciiArt for nodes
 - Nodes
 - () or (p)
 - Labels, tags
 - (p:Person:Mammal)
 - Properties
 - (p:Person { name: 'John'})
- AsciiArt for relationship
 - Relationship
 - --> -[a:ACTED_IN]->
 - Direction
 - (p1) -[:ACTED_IN]-> (p2) (p1) <-[:ACTED_IN]- (p2)
 - Properties
 - (p1) -[a:ACTED_IN { type: 'series' }]-> (p2)
- Aliases
 - p1, p2, a

Cypher

■ Actors

- Marek Kondrat (MK)
 - Ur. 18.10.1950
- Piotr Fronczewski (PF)
 - Ur. 8.6.1946
- Krzysztof Kowalewski (KK)
 - Ur. 20.03.1937
- Janusz Gajos (JG)
 - Ur. 23.09.1939
- Zbigniew Zapasiewicz (ZZ)
 - Ur. 13.09.1934, Zm. 14.07.2009

■ Movies

- Psy (Psy)
- C.K. Dezerterzy (CKD)
- Dzień świra (DS)
- Miś (Miś)
- Akademia Pana Kleksa (APK)

■ Acted in

- Psy
 - MK (Olo)
 - ZZ (Wencel)
 - JG (Siwy)
- CKD
 - MK (Kania)
 - KK (boss)
 - ZZ (Wagner)
- DS
 - MK (Adaś Miauczyński)
 - PF (doctor)
- Miś
 - KK (Jan Hochwander)
- APK
 - PF (Ambroży Kleks)

Cypher

- **CRUD Examples**

```
CREATE (:Person { name: "Marek Kondrat" }) -[:ACTED_IN]-> (:Movie { name: "Psy" })
CREATE (:Person { name: "Janusz Gajos", born: "23.09.1939" })
MATCH (:Movie { name: "Psy" })<-[:ACTED_IN]-(:Person { name: "Marek Kondrat"}) SET
a.character="Olo" RETURN a
```

```
MATCH (p:Person),(m:Movie)
WHERE p.name = 'Janusz Gajos' AND m.name = 'Psy'
CREATE (p)-[:ACTED_IN { character: "Siwy" }]->(m)
RETURN type(r), r.character
```

```
MATCH (:Movie { name: "Psy" })<-[:ACTED_IN]-(:Person) RETURN p
MATCH (m:Movie)<-[:ACTED_IN]-(:Person) WHERE m.name="Psy" RETURN p
```

```
MATCH (n:Person { name: 'UNKNOWN' })
DELETE n
```

```
MATCH (n)
DETACH DELETE n
```

Więcej: <https://neo4j.com/docs/cypher-manual/current/clauses/>

Others

- Potentially useful
 - <https://neo4j.com/docs/operations-manual/current/configuration/password-and-user-recovery/>