**RDBMS ↔ Graph (RDF/Neo4j) — Updated with Neo4j Concepts**

Generated: 2025-08-31 21:19 UTC. This text-only document compares a small car domain modeled in a relational database and as a graph, and highlights Neo4j-specific concepts (relationship type & direction, node labels, properties).

**1) Concept map (what equals what)**

|  |  |
| --- | --- |
| RDBMS | Graph (RDF / Neo4j property graph) |
| Table | Class / Node label (Neo4j nodes can have 0..n labels) |
| Row | Node (instance) |
| Column | Property (on node or relationship) |
| Foreign key + JOIN | Typed, directed relationship + pattern match |
| Constraints (PK/FK/CHECK) | Neo4j constraints (indexes/uniqueness); SHACL for RDF shapes |
| SQL | SPARQL (RDF) / Cypher (Neo4j) |

Neo4j specifics: A relationship \*\*must have exactly one relationship type\*\* and \*\*always has a direction\*\*. Both nodes and relationships can have \*\*properties\*\*. Nodes can have \*\*zero or more labels\*\*.

**2) RDBMS tables — two-row sample**

TABLE: vehicle\_model  
+----+------------+-----------+  
| id | code | name |  
+----+------------+-----------+  
| M1 | FALCON\_X | Falcon X |  
| M2 | ROADSTER | Roadster |  
+----+------------+-----------+  
  
TABLE: trim  
+------+----------+-----------+  
| id | model\_id | model\_year|  
+------+----------+-----------+  
| T101 | M1 | 2023 |  
| T102 | M1 | 2023 |  
+------+----------+-----------+  
  
TABLE: feature  
+------+-------+---------------------------+  
| id | code | pref\_label |  
+------+-------+---------------------------+  
| F1 | ACC | Adaptive Cruise Control |  
| F2 | HUD | Head-Up Display |  
+------+-------+---------------------------+  
  
TABLE: trim\_feature (N:M bridge)  
+--------+-----------+  
| trim\_id| feature\_id|  
+--------+-----------+  
| T101 | F1 |  
| T102 | F2 |  
+--------+-----------+  
  
TABLE: region  
+----+------+-----------+  
| id | code | name |  
+----+------+-----------+  
| CA | CA | California|  
| TX | TX | Texas |  
+----+------+-----------+  
  
TABLE: sale\_record  
+----+--------+----------+------------+----------+  
| id | trim\_id| region\_id| sale\_date | quantity |  
+----+--------+----------+------------+----------+  
| S1 | T101 | CA | 2023-03-15 | 120 |  
| S2 | T102 | CA | 2023-03-15 | 55 |  
+----+--------+----------+------------+----------+

**3) Graph schema — labels/classes, properties, and relationship types (directed)**

(:VehicleModel) <-[:BELONGS\_TO]- (:Trim) -[:HAS\_FEATURE]-> (:Feature)  
 | |  
 | code : string (property) | modelYear : integer (property)  
 v v  
  
(:SaleRecord) -[:FOR\_TRIM]-> (:Trim)  
 | \  
 | \-[:SOLD\_IN]-> (:Region)  
 |  
 +-- saleDate : date (property)  
 +-- quantity : integer (property)  
  
(:Region)  
 |  
 +-- code : string (property)

**4) Graph instances — same facts in Neo4j and RDF**

Neo4j (labels, properties, typed & directed relationships):

(:VehicleModel {code:"FALCON\_X"}) <-[:BELONGS\_TO]- (:Trim {code:"T101", modelYear:2023})  
 |  
 +--[:HAS\_FEATURE]-> (:Feature {code:"ACC", prefLabel:"Adaptive Cruise Control"})  
  
(:VehicleModel {code:"FALCON\_X"}) <-[:BELONGS\_TO]- (:Trim {code:"T102", modelYear:2023})  
 +--[:HAS\_FEATURE]-> (:Feature {code:"HUD", prefLabel:"Head-Up Display"})  
  
(:SaleRecord {id:"S1", saleDate:date("2023-03-15"), quantity:120})-[:FOR\_TRIM]->(:Trim {code:"T101"})  
(:SaleRecord {id:"S1"})-[:SOLD\_IN]->(:Region {code:"CA", name:"California"})  
  
(:SaleRecord {id:"S2", saleDate:date("2023-03-15"), quantity:55})-[:FOR\_TRIM]->(:Trim {code:"T102"})  
(:SaleRecord {id:"S2"})-[:SOLD\_IN]->(:Region {code:"CA"})

RDF/Turtle (subject → predicate → object; same directionality):

@prefix ex: <http://example.com/auto#> .  
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .  
  
ex:M1 a ex:VehicleModel ; ex:code "FALCON\_X" .  
  
ex:T101 a ex:Trim ; ex:belongsTo ex:M1 ; ex:modelYear "2023"^^xsd:gYear ;  
 ex:hasFeature ex:F\_ACC .  
  
ex:T102 a ex:Trim ; ex:belongsTo ex:M1 ; ex:modelYear "2023"^^xsd:gYear ;  
 ex:hasFeature ex:F\_HUD .  
  
ex:F\_ACC a ex:Feature ; ex:prefLabel "Adaptive Cruise Control" .  
ex:F\_HUD a ex:Feature ; ex:prefLabel "Head-Up Display" .  
  
ex:S1 a ex:SaleRecord ; ex:forTrim ex:T101 ; ex:soldIn ex:CA ;  
 ex:saleDate "2023-03-15"^^xsd:date ; ex:quantity 120 .  
  
ex:S2 a ex:SaleRecord ; ex:forTrim ex:T102 ; ex:soldIn ex:CA ;  
 ex:saleDate "2023-03-15"^^xsd:date ; ex:quantity 55 .

**5) Ask the same question — join path vs traversal path**

CQ: Which 2023 Falcon X trims have ACC and California sales on 2023‑03‑15, and how many units?

RDBMS (JOIN path):  
vehicle\_model(code='FALCON\_X')  
 └─(id)=model\_id──> trim(model\_year=2023)  
 ├─(id)=trim\_id──> trim\_feature ──(feature\_id)=id──> feature(code='ACC')  
 └─(id)=trim\_id──> sale\_record(sale\_date='2023-03-15', region\_id='CA')  
Aggregate: SUM(sale\_record.quantity) GROUP BY trim\_id

Graph (Traversal path with typed & directed relationships):  
( VehicleModel code="FALCON\_X" )  
 ^<-[:BELONGS\_TO]- ( Trim modelYear=2023 ) -[:HAS\_FEATURE]-> ( Feature code/prefLabel ~ "ACC" )  
 |  
 +-[:FOR\_TRIM]- ( SaleRecord saleDate=2023-03-15, quantity ) -[:SOLD\_IN]-> ( Region code="CA" )  
Aggregate: SUM(quantity) GROUP BY Trim

**6) Neo4j essentials (used above)**

• Relationships connect a source node to a target node, must have exactly one relationship type, and always have a direction. Both nodes and relationships can have properties. Nodes can have zero or more labels. Indexes and constraints are optional but recommended for performance and data quality.