

## Tutorial 10a

### Neo4j – Truck Delivery

# SOLUTIONS

### Truck Delivery Case Study

A truck company is responsible for picking up shipments from warehouses of a retail chain called MYER and delivering the shipments to individual retail store of MYER. A truck may carry several shipments during a single trip and delivers those shipments to multiple stores. Trucks have different capacities for both the volumes they can hold and the weights they can carry. At the moment, a truck makes several trips each week. A relational database is being used to keep track of the deliveries, including the scheduling of trucks, which provide timely deliveries to stores. The data are shown in Table 1-6.

**Table 1. Warehouse**

WAREHOUSE_ID	LOCATION
W1	Warehouse1
W2	Warehouse2
W3	Warehouse3
W4	Warehouse4
W5	Warehouse5

**Table 2. Truck**

TRUCK_ID	VOL_CAPACITY	WEIGHT_CATEGORY	COST_PER_KM
Truck1	250	Medium	1.2
Truck2	300	Medium	1.5
Truck3	100	Small	0.8
Truck4	550	Large	2.3
Truck5	650	Large	2.5

**Table 3. Trip**

TRIP_ID	TRIP_DATE	TOTAL_KM	TRUCK_ID
Trip1	14-Apr-13	370	Truck1
Trip2	14-Apr-13	570	Truck2
Trip3	14-Apr-13	250	Truck3
Trip4	15-Apr-13	450	Truck1
Trip5	15-Apr-13	175	Truck2

**Table 4. Trip\_From**

TRIP_ID	WAREHOUSE_ID
Trip1	W1
Trip1	W4
Trip1	W5
Trip2	W1
Trip2	W2

Trip3	W1
Trip3	W5
Trip4	W1
Trip5	W4
Trip5	W5

**Table 5. Store**

STORE_ID	STORE_NAME	STORE_ADDRESS
M1	Myer City	Melbourne
M2	Myer Chaddy	Chadstone
M3	Myer Highpoint	High Point
M4	Myer West	Doncaster
M5	Myer North	Northland
M6	Myer South	Southland
M7	Myer East	Eastland
M8	Myer Knox	Knox

**Table 6. Destination**

TRIP_ID	STORE_ID
Trip1	M1
Trip1	M2
Trip1	M4
Trip1	M3
Trip1	M8
Trip2	M4
Trip2	M1
Trip2	M2

However, the company would like to change their current relational database to a graph database. Thus they have asked you to design a new graph database that can store the data as shown in Table 1-6. The data are also available in CSV files provided in the Moodle site.

**Tasks:**


1. List all the entities and relationships for the Truck Delivery case study.

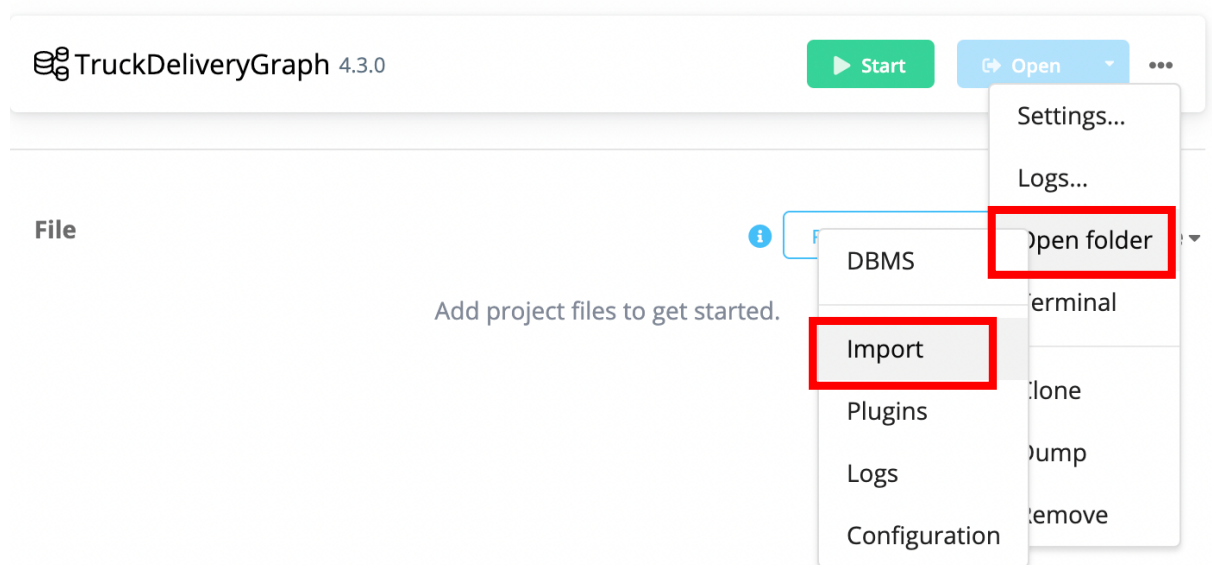
**Entities:**

- Warehouse
- Truck
- Store
- Trip

**Relationships:**

- Trip\_From
- Destination

2. Create a new project in Neo4j and name it as Tutorial10a.
3. Add a new graph. Name the graph to TruckDeliveryGraph and set the password to graph.
4. After your graph has been created, start the graph then open Neo4j Browser.
5. Download the CSV files available on Moodle.
6. Go back to your Project page in Neo4j, click on the  option on the top right corner, and then select Open folder.
7. Click on the drop-down menu beside the Open Folder button and select Import



8. The Import folder will be opened for you. Paste the CSV files into this folder so that we can import these files into Neo4j.
9. Open Neo4j Browser.
10. Let's create new nodes that represent the warehouses (Table 1) by loading the CSV file. The code is as follow.

```
LOAD CSV WITH HEADERS FROM "file:///warehouse.csv"
AS row
WITH row WHERE row.warehouse_id IS NOT NULL
MERGE (w:Warehouses {warehouse_id: row.warehouse_id})
ON CREATE SET w.location = row.location;
```

11. Import the other nodes, following the command from previous step.

```
LOAD CSV WITH HEADERS FROM "file:///truck.csv"
```

```

AS row
WITH row WHERE row.truck_id IS NOT NULL
MERGE (t:Trucks {truck_id: row.truck_id})
ON CREATE SET t.vol_capacity = row.vol_capacity,
               t.weight_category = row.weight_category,
               t.cost_per_km = row.cost_per_km;

LOAD CSV WITH HEADERS FROM "file:///store.csv"
AS row
WITH row WHERE row.store_id IS NOT NULL
MERGE (s:Stores {store_id: row.store_id})
ON CREATE SET s.store_name = row.store_name,
               s.store_address = row.store_address;

LOAD CSV WITH HEADERS FROM "file:///trip.csv"
AS row
WITH row WHERE row.trip_id IS NOT NULL
MERGE (t:Trips {trip_id: row.trip_id})
ON CREATE SET t.trip_date = row.trip_date,
               t.total_km = row.total_km,
               t.truck_id = row.truck_id;

```

## 12. Create the relationships.

```

LOAD CSV WITH HEADERS FROM "file:///trip_from.csv" AS
csvLine
MATCH (t:Trips {trip_id: csvLine.trip_id})
MATCH (w:Warehouses {warehouse_id: csvLine.warehouse_id})
CREATE (t)-[:PICKUP_FROM]->(w);

```

## 13. Import the remaining relationships to complete graph.

```

LOAD CSV WITH HEADERS FROM "file:///destination.csv" AS
csvLine
MATCH (t:Trips {trip_id: csvLine.trip_id})
MATCH (s:Stores {store_id: csvLine.store_id})
CREATE (t)-[:DELIVER_TO]->(s);

MATCH (trip:Trips), (truck:Trucks)
WHERE (EXISTS (trip.truck_id) OR EXISTS (truck.trip_id))
AND trip.truck_id = truck.truck_id
MERGE (truck)-[:USED_IN]->(trip);

```

## 14. Answer the following queries.

### 1. Display all the trucks.

```

MATCH (n:Trucks)

```

```
RETURN n
```

2. How many warehouses are visited in Trip1?

```
MATCH (w:Warehouses)--(t:Trips {trip_id: "Trip1"})
RETURN count(w)
```

3. Show trucks that have not been used yet.

```
MATCH (truck:Trucks)
WHERE NOT (truck)--()
RETURN truck
```

4. Which trip picks up shipments from warehouse W2 and delivers to Myer City?

```
MATCH (w:Warehouses {warehouse_id:"W1"})--(n)--(s:Stores
{store_id:"M1"})
RETURN n
```

5. Display all warehouses and stores connected to Trip3.

```
MATCH (w)--(trip:Trips {trip_id:"Trip3"})--(s)
RETURN w, trip, s
```

**The End**