Large Scale Data Management
NoSQL Systems and Data Models

Exam

January 29th, 2026

Part A – Course Questions (2 points)

Write the single best answer for each question on the answer sheet (not on the subject).

1. *In the context of Big Data, which "V" describes the challenge of handling data that arrives in continuous streams and requires real-time processing?*
   A. Volume
   B. Velocity
   C. Variety
   D. Veracity

2. *You are designing a distributed system that must function even if network communication between servers fails. According to the CAP theorem, which two properties can you choose to prioritize?*
   A. Consistency and Availability
   B. Consistency and Partition Tolerance
   C. Availability and Partition Tolerance
   D. You must choose either CP or AP (Consistency + Partition Tolerance or Availability + Partition Tolerance).

3. *Why are NoSQL Key-Value stores generally faster for simple lookups compared to Relational Databases?*
   A. They use complex joins to pre-calculate data.
   B. They typically avoid enforcing strict schemas and referential integrity checks on every write.
   C. They store data exclusively in RAM and never on disk.
   D. They use SQL which is optimized for key retrieval.

4. *When using a graph database for navigating in graphs, what is the primary performance advantage over a relational database?*
   A. Graph databases use smaller hard drives.
   B. Graph databases use adjacency pointers, avoiding expensive `JOIN` operations.
   C. Graph databases do not support properties on relationships, saving space.
   D. Graph databases enforce 3rd normal form automatically.

Subject continues on next page →

Part B – OLAP Database Design (3 points)

Scenario "GlobalStay" is an online platform for booking vacation rentals. The management team wants to build a Data Warehouse to analyze revenue performance. The system records every completed booking. Management specifically needs to analyze the **Total revenue** and **Duration of stay (in days)**.

They want to analyze this data based on: *when* the booking happened (day resolution, but aggregating up to month, quarter, and year); *where* the property is located (city, region, and country); *who* the host is (host ID and name, and their status, Superhost or not); *what* type of property was booked (property type, e.g., *Apartment*, *Villa*, and number of rooms).

Questions Answer the following questions regarding the multidimensional modelization of this scenario:

1. Design the schema of the fact table. List the foreign keys and the measures (facts) that should be contained in this table.
2. How many dimensions does your modelisation have? For each dimension, specify the full hierarchy required, and clearly state the number of levels needed.
3. Sketch the relational schema used to support the multidimensional modelization above. Is it a start or snowflake schema? Justify your choice briefly based on how you handled the dimensions.

Part C – Graph Databases and Cypher (3 points)

Graph Consider the following property graph representing a **Global Supply Chain**, with the following rough schema:

- Node types: `(:Factory {name, city})`; `(:Part {id, name, cost})`; `(:Product {name, retail_price})`
- Relationships: `(:Factory)-[:MANUFACTURES {year: xxxx}]->(:Part)`; `(:Part)-[:COMPONENT_OF {quantity_needed}]->(:Product)`

The example graph is given in Figure 1 and the full data in Cypher format is provided in the Appendix.

Question 1 Explain clearly what the following Cypher query outputs. Describe the logic and the final output.

```
MATCH (f:Factory)-[:MANUFACTURES]->(p:Part)-[r:COMPONENT_OF]->(prod:Product)
  WHERE f.city = 'Berlin' AND prod.retail_price > 1000
  WITH prod, sum(p.cost * r.quantity_needed) AS production_cost
RETURN prod.name, production_cost, (prod.retail_price - production_cost) AS margin
ORDER BY margin DESC
```

Subject continues on next page →

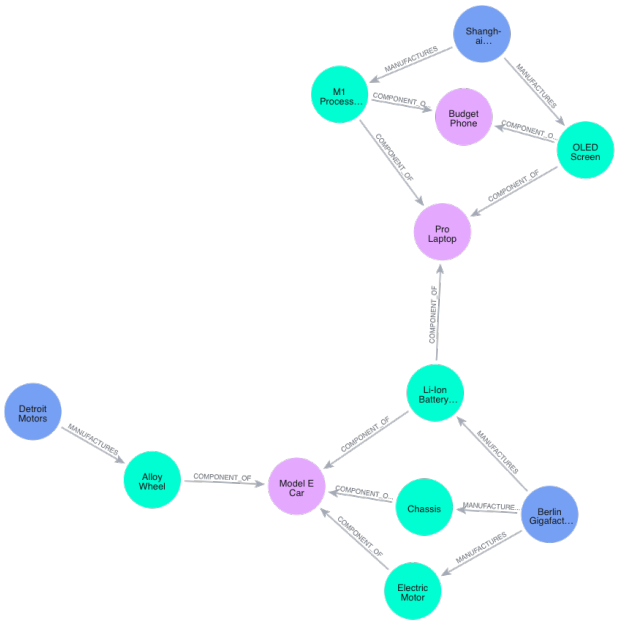

Figure 1: Example Global Supply Chain graph.

Question 2 The company realizes that a specific `Part` (e.g., battery) might be compatible with multiple `Products`, but the *quantity required* differs (e.g., a phone needs 1 battery, a laptop needs 2).

In the schema provided above, where is this specific information currently stored?

If we wanted to track *defective* parts returned by customers, would it be better to add a property `defective_count` to the `Part` node, or create a new node `(:Return)` connected to the `Part`? Explain your reasoning.

Part D – System Design (2 points)

Specification You are the Lead Data Engineer for a large software company. Users often submit support tickets with vague descriptions, such as *"My screen went dark when I closed the lid"* or *"The computer sleeps when folded."* Your existing database uses standard SQL keyword search (e.g., `WHERE text LIKE '%screen%'`), but it fails to find the relevant technical manual article titled *"Lid Close Laptop Settings"* because the user didn't use the exact word "*screen*" in one of the descriptions.

Question Propose a data solution to solve this problem. Identify the specific type of database technology (covered in the course) best suited for this non-exact search. Explain the process of how you would transform the text of the manual articles and the user's query into a format this database can understand. Describe how the system determines which article is the "best" match for the user's query.

Describe your solution in detail, but concisely, focusing on the key components and steps involved. The description should be at most three paragraphs in length.

End of subject.

Full Graph Data for Part C

Nodes

```
(:Factory {city: "Berlin", name: "Berlin Gigafactory"})
(:Factory {city: "Shanghai", name: "Shanghai Assembly"})
(:Factory {city: "Detroit", name: "Detroit Motors"})

(:Part {cost: 3000, name: "Li-Ion Battery Pack", id: "P001"})5
(:Part {cost: 1500, name: "Electric Motor", id: "P002"})
(:Part {cost: 200, name: "Alloy Wheel", id: "P003"})
(:Part {cost: 150, name: "OLED Screen", id: "P004"})
(:Part {cost: 300, name: "M1 Processor", id: "P005"})
(:Part {cost: 500, name: "Chassis", id: "P006"})

(:Product {name: "Model E Car", retail_price: 45000})
(:Product {name: "Pro Laptop", retail_price: 2500})
(:Product {name: "Budget Phone", retail_price: 400})
```

Relationships

```
(:Factory {city: "Berlin", name: "Berlin Gigafactory"})-[:MANUFACTURES {year: 2023}]->
  (:Part {cost: 3000, name: "Li-Ion Battery Pack", id: "P001"})
(:Factory {city: "Berlin", name: "Berlin Gigafactory"})-[:MANUFACTURES {year: 2023}]->
  (:Part {cost: 1500, name: "Electric Motor", id: "P002"})
(:Factory {city: "Berlin", name: "Berlin Gigafactory"})-[:MANUFACTURES {year: 2022}]->
  (:Part {cost: 500, name: "Chassis", id: "P006"})
(:Factory {city: "Shanghai", name: "Shanghai Assembly"})-[:MANUFACTURES {year: 2023}]->
  (:Part {cost: 150, name: "OLED Screen", id: "P004"})
(:Factory {city: "Shanghai", name: "Shanghai Assembly"})-[:MANUFACTURES {year: 2023}]->
  (:Part {cost: 300, name: "M1 Processor", id: "P005"})
(:Factory {city: "Shanghai", name: "Shanghai Assembly"})-[:MANUFACTURES {year: 2022}]->
  (:Part {cost: 3000, name: "Li-Ion Battery Pack", id: "P001"})
(:Factory {city: "Detroit", name: "Detroit Motors"})-[:MANUFACTURES {year: 2023}]->
  (:Part {cost: 200, name: "Alloy Wheel", id: "P003"})

(:Part {cost: 3000, name: "Li-Ion Battery Pack", id: "P001"})-[:COMPONENT_OF {quantity_needed: 1}]->
  (:Product {name: "Model E Car", retail_price: 45000})
(:Part {cost: 3000, name: "Li-Ion Battery Pack", id: "P001"})-[:COMPONENT_OF {quantity_needed: 1}]->
  (:Product {name: "Pro Laptop", retail_price: 2500})
(:Part {cost: 1500, name: "Electric Motor", id: "P002"})-[:COMPONENT_OF {quantity_needed: 1}]->
  (:Product {name: "Model E Car", retail_price: 45000})
(:Part {cost: 200, name: "Alloy Wheel", id: "P003"})-[:COMPONENT_OF {quantity_needed: 4}]->
  (:Product {name: "Model E Car", retail_price: 45000})
(:Part {cost: 150, name: "OLED Screen", id: "P004"})-[:COMPONENT_OF {quantity_needed: 1}]->
  (:Product {name: "Pro Laptop", retail_price: 2500})
(:Part {cost: 150, name: "OLED Screen", id: "P004"})-[:COMPONENT_OF {quantity_needed: 1}]->
  (:Product {name: "Budget Phone", retail_price: 400})
(:Part {cost: 300, name: "M1 Processor", id: "P005"})-[:COMPONENT_OF {quantity_needed: 1}]->
  (:Product {name: "Pro Laptop", retail_price: 2500})
(:Part {cost: 300, name: "M1 Processor", id: "P005"})-[:COMPONENT_OF {quantity_needed: 1}]->
  (:Product {name: "Budget Phone", retail_price: 400})
(:Part {cost: 500, name: "Chassis", id: "P006"})-[:COMPONENT_OF {quantity_needed: 1}]->
  (:Product {name: "Model E Car", retail_price: 45000})
```