Lab 6 - Graph Databases: Neo4j

1. Chosen Structure for Neo4j Database

Graph Schema Design:

The database schema is adapted from the TPC-H benchmark. The graph consists of nodes representing entities and relationships capturing connections between them. This design is optimized to improve the performance of the four provided queries.

Nodes and Properties:

1. **Region:**
   - Properties: {regionKey, name, comment}
   - Cardinality: Only 5 regions exist, making this node type lightweight and efficient.
   - Justification: regionKey is indexed to support fast filtering for regional queries (e.g., Query 2 and Query 4). Edges connect to Nation nodes for efficient traversal.
2. **Nation:**
   - Properties: {nationKey, name, regionKey, comment}
   - Cardinality: 25 nations.
   - Justification: A BELONGS_TO relationship links Nation to Region, reducing query complexity for filtering nations within a specific region.
3. **Customer:**
   - Properties: {custKey, name, address, nationKey, phone, acctBal, mktSegment, comment}
   - Cardinality: ~150,000 customers.
   - Justification: Queries often filter customers by mktSegment (Query 3). Indexing this property speeds up segmentation-based filtering.
4. **Supplier**:
   - Properties: {suppKey, name, address, nationKey, phone, acctBal, comment}
   - Cardinality: ~10,000 suppliers.
   - Justification: Relationships such as LOCATED_IN to Nation and SUPPLIES to PartSupplier provide efficient navigation for Query 2.
5. **Part**:
   - Properties: {partKey, name, mfgr, brand, type, size, container, retailPrice, comment}
   - Cardinality: ~200,000 parts.
   - Justification: Query 2 filters parts by type and size. Indexing these properties ensures fast lookups.

6. **PartSupplier**:
    - Properties: {partSuppKey, partKey, suppKey, availQty, supplyCost, comment}
    - Cardinality: ~800,000 relationships between parts and suppliers.
    - Justification: Storing supplyCost directly on this node reduces the complexity of identifying minimal supply costs in Query 2.
7. **Order**:
    - Properties: {orderKey, custKey, orderStatus, totalPrice, orderDate, orderPriority, clerk, shipPriority, comment}
    - Cardinality: ~1.5M orders.
    - Justification: Relationships such as PLACED (to Customer) and CONTAINS (to LineItem) support complex aggregations in Query 3 and Query 4.
8. **LineItem**:
    - Properties: {lineNumber, orderKey, quantity, extendedPrice, discount, tax, returnFlag, lineStatus, shipDate, commitDate, receiptDate, shipInstruct, shipMode, comment, suppKey}
    - Cardinality: ~6M line items.
    - Justification: Critical for Query 1 and Query 4 aggregations. Properties such as shipDate, extendedPrice, and discount are indexed for quick filtering and summation.

Relationships:

1. **BELONGS_TO**:
    - Links Nation to Region and Customer to Nation.
    - Quantitative Impact: Allows direct filtering of nations or customers by region in O(1) traversal.
2. **LOCATED_IN**:
    - Links Supplier to Nation.
    - Quantitative Impact: Simplifies supplier filtering by geography in Query 2.
3. **SUPPLIED_WITH**:
    - Links Part to PartSupplier.
    - Quantitative Impact: Reduces traversal complexity for part-supplier relationships in Query 2.
4. **SUPPLIES**:
    - Links PartSupplier to Supplier.
    - Quantitative Impact: Enables efficient cost comparison for suppliers in Query 2.
5. **PLACED**:
    - Links Customer to Order.
    - Quantitative Impact: Facilitates rapid aggregation of customer order data for Query 3.
6. **CONTAINS**:
    - Links Order to LineItem.

- Quantitative Impact: Reduces lookup complexity for line items in orders, critical for Query 1 and Query 4.

7. **SUPPLIED_BY**:
   - Links LineItem to Supplier.
   - Quantitative Impact: Directly associates line items with suppliers, reducing multi-step lookups in Query 4.

Indexing Strategy:

1. **Unique Constraints**:
   - Enforced on keys such as regionKey, nationKey, custKey, suppKey, partKey, orderKey, lineNumber, and partSuppKey.
2. **Additional Indexes**:
   - Frequently queried attributes are indexed:
     - Region.name: Speeds up filtering for specific regions.
     - Order.orderDate: Essential for date-range queries (e.g., Query 4).
     - LineItem.shipDate: Crucial for filtering line items in Query 1 and Query 3.
     - Part.size and Part.type: Optimizes part lookups in Query 2.
3. **Quantitative Impact**:
   - Indexing reduces query time complexity from O(n) to O(log n) for indexed properties, enabling efficient query execution on datasets with millions of records.

2. Cypher Query Definitions

Query 1: Aggregate Metrics for Line Items

```
MATCH (l:LineItem)
WHERE l.shipDate <= $date
RETURN
  l.returnFlag AS returnFlag,
  l.lineStatus AS lineStatus,
  SUM(l.quantity) AS sum_qty,
  SUM(l.extendedPrice) AS sum_base_price,
  SUM(l.extendedPrice * (1 - l.discount)) AS sum_disc_price,
  SUM(l.extendedPrice * (1 - l.discount) * (1 + l.tax)) AS sum_charge,
  AVG(l.quantity) AS avg_qty,
  AVG(l.extendedPrice) AS avg_price,
  AVG(l.discount) AS avg_disc,
  COUNT(*) AS count_order
ORDER BY returnFlag, lineStatus;
```

Query 2: Supplier Selection by Part Type and Region

```
MATCH (p:Part)-[:SUPPLIED_WITH]->(ps:PartSupplier)-[:SUPPLIES]->(s:Supplier),
     (s)-[:LOCATED_IN]->(n:Nation)-[:BELONGS_TO]->(r:Region)
WHERE p.size = $size AND p.type CONTAINS $type AND r.name = $region
WITH p, ps, s, MIN(ps.supplyCost) AS min_cost
WHERE ps.supplyCost = min_cost
RETURN
  s.acctBal AS acctBal,
  s.name AS supplierName,
  n.name AS nationName,
  p.partKey AS partKey,
  p.mfgr AS manufacturer,
  s.address AS address,
  s.phone AS phone,
  s.comment AS supplierComment
ORDER BY acctBal DESC, nationName, supplierName, partKey;
```

Query 3: Order Revenue by Customer Segment

```
MATCH (c:Customer)-[:PLACED]->(o:Order)-[:CONTAINS]->(l:LineItem)
WHERE c.mktSegment = $segment AND o.orderDate < $date1 AND l.shipDate >
$date2
RETURN
  l.orderKey AS orderKey,
  SUM(l.extendedPrice * (1 - l.discount)) AS revenue,
  o.orderDate AS orderDate,
  o.shipPriority AS shipPriority
ORDER BY revenue DESC, orderDate;
```

Query 4: Revenue by Region and Nation

```
MATCH
(c:Customer)-[:PLACED]->(o:Order)-[:CONTAINS]->(l:LineItem)-[:SUPPLIED_BY]->(s:
Supplier)-[:LOCATED_IN]->(n:Nation)-[:BELONGS_TO]->(r:Region)
WHERE r.name = $region AND date(o.orderDate) >= date($start_date) AND
date(o.orderDate) < date($start_date) + duration('P1Y')
RETURN
  n.name AS nationName,
  SUM(l.extendedPrice * (1 - l.discount)) AS revenue
ORDER BY revenue DESC;
```