ENSF 608 – Databases LAB Session (SEP. 19th)
Dr. Ronnie de Souza Santos, PhD.

Scenario Description:

Calgary Superstore is one of the largest retail chains in Alberta. Each **branch** of the superstore manages its own products, employees, customers, and daily operations, while reporting to a single **head office** that oversees the entire company. Management has decided to adopt a **hierarchical database system** to organize and store its information in a way that reflects the real-world structure of the business. The data inside the database mirrors what happens daily in the store:

- **At the top is the SUPERSTORE**, representing the company as a whole. From here, management wants to monitor every branch, compare performance, and ensure consistency across all locations.
- **Each SUPERSTORE has multiple BRANCHES**, located in different parts of Calgary: "Downtown," "North Calgary," and "South Calgary"). Each branch is semi-autonomous and has its own staff, inventory, and customers.
- **Within each BRANCH, there are several DEPARTMENTS**, such as:
    - **Electronics**: TVs, laptops, phones, headphones.
    - **Groceries**: fresh produce, frozen meals, dairy, bread.
    - **Clothing**: men's, women's, and children's apparel.
    - **Furniture**: beds, sofas, tables, storage cabinets.

Each department must manage its own product line and stock levels. For example, the Electronics department may run a weekend promotion on smartphones, while Groceries may have to deal with a sudden shortage of fresh vegetables.

- **Each DEPARTMENT sells PRODUCTS.** A product record needs details such as a product identifier, description, price, supplier, and the number of units available. For example, the Clothing department may have "winter jackets" that sell quickly in December, while the Groceries department tracks expiration dates on milk cartons to avoid waste.
- **Each BRANCH employs multiple EMPLOYEES**, with roles like store managers, cashiers, sales associates, and warehouse workers. Each employee record includes identifiers, names, positions, salaries, and hiring dates. For instance, the Electronics department might have a specialist who provides customer advice, while the checkout area needs several cashiers during peak hours.
- **Each BRANCH also serves many CUSTOMERS.** Customers may be casual shoppers or part of a loyalty program with membership statuses such as Regular, Silver, Gold, or Premium. Customer records include identifiers, names, ages, contact details, and membership level. For example, a Premium member might get discounts on bulk grocery purchases, while a Regular customer pays full price.
- **CUSTOMERS place ORDERS** during their visits. An order record contains an order ID, the date, the total amount, and a list of purchased items. For example, a single order might include a laptop from Electronics, milk from Groceries, and a sofa from Furniture, all under

the same transaction. Orders are also linked to customers for tracking purchase history and offering targeted promotions.

- **Each BRANCH maintains INFRASTRUCTURE** that keeps the store running smoothly. This includes storage warehouses for goods, checkout counters where payments are processed, delivery trucks for transporting furniture, and even in-store bakeries or restaurants in some locations. For example, if a delivery truck breaks down, the Furniture department may not be able to complete scheduled customer deliveries.

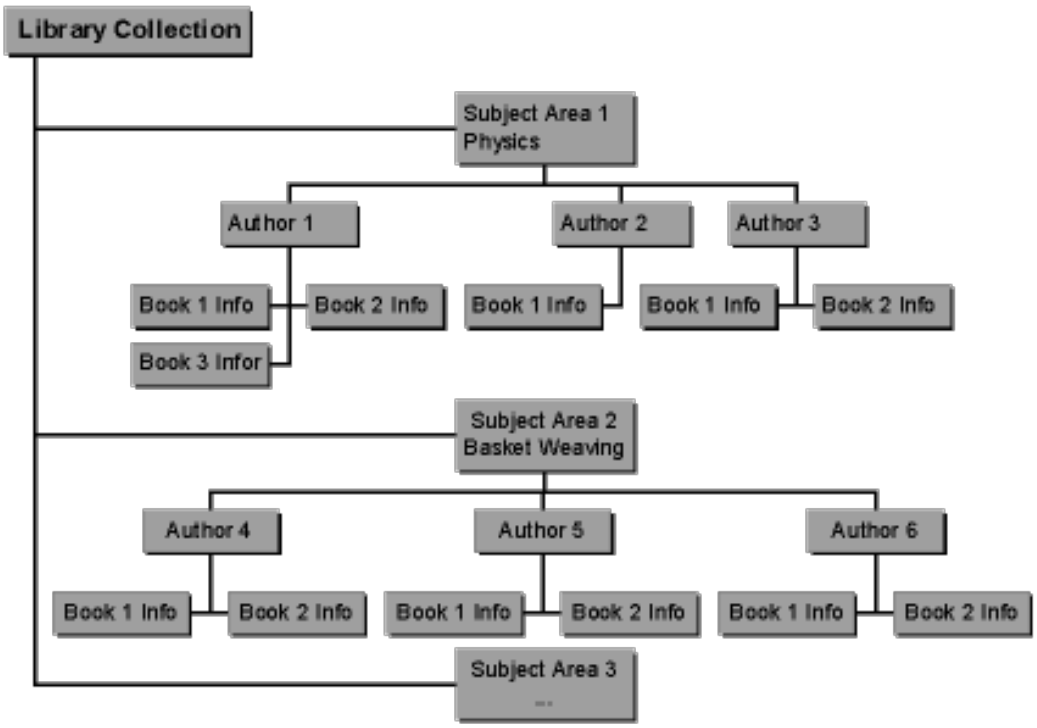
Task 1 – Identify Entities and Attributes

Read the scenario carefully and decide:

- Which entities should appear in the hierarchical database (e.g., the top-level entity, sub-entities, child entities).
- For each entity, list at least 3–5 attributes (e.g., *Entity 1: Attribute 1, Attribute 2, Attribute 3*).
- Remember: attributes should describe the entity (e.g., for a customer: ID, name, age).

Task 2 – Build the Block Diagram

- Draw a block diagram where each block represents an entity and lists its attributes like the example below.
- Connect the blocks with arrows to represent parent–child relationships.
- Do not copy the examples from the scenario, instead build your own structure from your analysis.



Task 3 – Write the Tree Structure

- Represent your database as a hierarchical tree in text form, like the example below.

```
COLLEGE: "University of Calgary"

 └── DEPARTMENT: "Schulich School of Engineering"
       ├── COURSE: "Software Engineering"
       │     ├── THEORY: (ID=1, Topic="Software Architecture", Hours=20)
       │     └── LABS:   (ID=1, Lab_Name="SE Lab 1", Capacity=30)
       │
       ├── COURSE: "Databases"
       │     ├── THEORY: (ID=2, Topic="SQL and Modeling", Hours=25)
       │     └── LABS:   (ID=2, Lab_Name="DB Lab 1", Capacity=25)
       │
       ├── TEACHER: (ID=501, Name="Prof. Souza", Specialization="Software Engineering")
       ├── TEACHER: (ID=502, Name="Prof. Santos", Specialization="Databases")
       │
       ├── STUDENT: (ID=101, Name="Alice", Age=20, Program="Software Engineering")
       ├── STUDENT: (ID=102, Name="Bob", Age=22, Program="Software Engineering")
       ├── STUDENT: (ID=201, Name="Carol", Age=21, Program="Databases")
       └── STUDENT: (ID=202, Name="Dan", Age=23, Program="Databases")

 └── INFRASTRUCTURE
       ├── (Infra_ID=1, Type="Library", Capacity=200)
       ├── (Infra_ID=2, Type="Main Building", Capacity=500)
       └── (Infra_ID=3, Type="Computer Labs", Capacity=150)
```

Task 4 – Define Navigational Queries

Describe how you would navigate the tree to answer the following:

1. List all items in one specific department of one specific branch.
2. Find all orders placed by one specific customer.
3. Retrieve all employees of one specific branch.

Use text to explain the path step by step (e.g., *Root → Branch → Department → Product*).

Task 5 – Reflection & Discussion

Using your database the management now wants to ask: *"Find all customers across all branches who purchased a specific product type."* Explain to them:

- Why is this type of query difficult in a hierarchical database?
- What workarounds could be used to support it?
- How would a relational database handle this more easily?