NAME - Shruti Prashant Lad shrutilad35@gmail.com

Part 2: Database Design

Objective

- Design a scalable, normalized database schema for a B2B inventory management system that:
- Supports multiple companies and warehouses
- Allows products to exist in multiple warehouses
- Tracks inventory changes over time
- Manages suppliers
- Supports bundled products

1. Database Schema Design

1.1 Companies

```
CREATE TABLE companies (
    id BIGSERIAL PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

1.2 Warehouses

```
CREATE TABLE warehouses (
    id BIGSERIAL PRIMARY KEY,
    company_id BIGINT NOT NULL,
    name VARCHAR(255) NOT NULL,
    location TEXT,
    FOREIGN KEY (company_id) REFERENCES companies(id)
);
```

Relationship:

- One company → many warehouses

1.3 Products

```
CREATE TABLE products (
    id BIGSERIAL PRIMARY KEY,
    company_id BIGINT NOT NULL,
    name VARCHAR(255) NOT NULL,
    sku VARCHAR(100) NOT NULL UNIQUE,
    price DECIMAL(10,2) NOT NULL,
    product_type VARCHAR(50),
    is_bundle BOOLEAN DEFAULT FALSE,
    low_stock_threshold INT,
    FOREIGN KEY (company_id) REFERENCES companies(id)
);
```

Notes:

- SKU is **globally unique**
- `is_bundle` identifies bundled products

1.4 Inventory

```
CREATE TABLE inventory (
    id BIGSERIAL PRIMARY KEY,
    product_id BIGINT NOT NULL,
    warehouse_id BIGINT NOT NULL,
    quantity INT NOT NULL DEFAULT 0,
    UNIQUE (product_id, warehouse_id),
    FOREIGN KEY (product_id) REFERENCES products(id),
    FOREIGN KEY (warehouse_id) REFERENCES warehouses(id)
);
```

Purpose:

- Resolves **many-to-many** relationship between products and warehouses

1.5 Inventory History (Audit Table)

```
CREATE TABLE inventory_history (
    id BIGSERIAL PRIMARY KEY,
    product_id BIGINT NOT NULL,
    warehouse_id BIGINT NOT NULL,
    change_quantity INT NOT NULL,
    reason VARCHAR(50),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (product_id) REFERENCES products(id),
    FOREIGN KEY (warehouse_id) REFERENCES warehouses(id)
);
```

Purpose:

- Tracks stock increases, decreases, adjustments, and audits

1.6 Suppliers

```
CREATE TABLE suppliers (
    id BIGSERIAL PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    contact_email VARCHAR(255)
);
```

1.7 Product–Supplier Mapping

```
CREATE TABLE product_suppliers (
    product_id BIGINT NOT NULL,
    supplier_id BIGINT NOT NULL,
    PRIMARY KEY (product_id, supplier_id),
    FOREIGN KEY (product_id) REFERENCES products(id),
    FOREIGN KEY (supplier_id) REFERENCES suppliers(id)
);
```

Purpose:

- Allows **many suppliers per product** and vice versa

1.8 Product Bundles

```
CREATE TABLE product_bundles (
    bundle_product_id BIGINT NOT NULL,
    component_product_id BIGINT NOT NULL,
    quantity INT NOT NULL,
    PRIMARY KEY (bundle_product_id, component_product_id),
    FOREIGN KEY (bundle_product_id) REFERENCES products(id),
    FOREIGN KEY (component_product_id) REFERENCES products(id)
);
```

Purpose:

- Represents bundled products (e.g., “Starter Kit”)

2. Missing Requirements / Questions for Product Team

1. Is SKU unique globally or per company?
2. Can a product have multiple suppliers with priority?
3. How is “recent sales activity” defined?
4. Can bundles be stocked independently?
5. Should inventory history be immutable?
6. Are soft deletes required?
7. Do products support multiple units (kg, pcs, liters)?
8. Is multi-currency pricing required?
9. Are suppliers company-specific or global?

3. Design Decisions & Justification

Why Separate Inventory Table

- Supports multiple warehouses
- Avoids product duplication
- Scales cleanly

Why Inventory History Table

- Required for audits
- Helps analytics and reporting
- Tracks stock movements

Indexes & Constraints

- `UNIQUE(sku)` → prevents duplicates
- `UNIQUE(product_id, warehouse_id)` → prevents duplicate inventory rows
- Foreign keys → maintain referential integrity

Why Bundle Mapping Table

- Flexible representation
- Supports nested bundles
- Avoids JSON-based storage (hard to query)

Conclusion

This database design:

- Follows normalization principles
- Supports scalability and auditing
- Aligns with real-world B2B SaaS inventory workflows
- Is easy to extend for analytics and reporting