CHRISSIE RAJ
G01465544

To keep track of vendors and products they supply, XYZ Corp. uses the table structure shown below. Assuming that the sample data are representative, draw a dependency diagram in Visio that shows all functional dependencies including both partial and transitive dependencies. (*Hint:* Look at the sample values to determine the nature of the relationships.)

PART_CODE	PART_DESC	VEND_NAME	VEND_ADDRESS	VEND_TYPE	PRICE
1234	Logic Chip	Fast Chips	Cupertino	Non-profit organization	25.00
1234	Logic Chip	Smart Chips	Phoenix	Profit organization	22.00
5678	Memory Chip	Fast Chips	Cupertino	Non-profit organization	18.00
5678	Memory Chip	Quality Chips	Austin	Profit organization	15.00

5678	Memory Chip	Smart Chips	Phoenix	Profit organization	19.00

2. Using the initial dependency diagram drawn in question 1, remove all partial dependencies, draw the new dependency diagrams in Visio, and identify the normal forms for each table structure you created.

3. Using the table structures you created in question 2, remove all transitive dependencies, and draw the new dependency diagrams in Visio. Also identify the normal forms for each table structure you created. If necessary, add or modify attributes to create appropriate determinants or to adhere to the naming conventions.

4. Using the results of question 3, draw the fully labeled Crow's Foot ERD in Visio. The diagram must include all entities, attributes, and relationships. Primary keys and foreign keys must be clearly identified on the diagram.

SOLUTION:

1: Initial Dependency Diagram – Identify Functional Dependencies

You are asked to create an **initial dependency diagram** showing all functional dependencies.

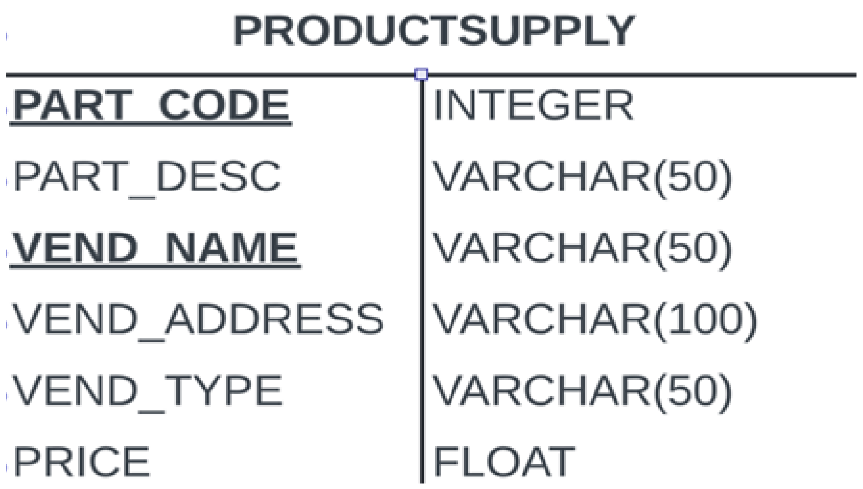
Identified Functional Dependencies:

1. **PART_CODE -> PART_DESC**: Each PART_CODE uniquely determines the description of the part (PART_DESC). This means if you know the PART_CODE, you can identify the PART_DESC.

2. **VEND_NAME -> VEND_ADDRESS, VEND_TYPE**: The name of the vendor (VEND_NAME) uniquely determines the vendor's address (VEND_ADDRESS) and the vendor type (VEND_TYPE). If you know the VEND_NAME, you can find its address and type.
3. **(PART_CODE, VEND_NAME) -> PRICE**: The combination of PART_CODE and VEND_NAME uniquely determines the PRICE. Since each vendor might offer the same part at a different price, you need both PART_CODE and VEND_NAME to uniquely determine the price.

Composite Primary Key:

- In the original unnormalized table, the **primary key** can be considered as a **composite key** (PART_CODE, VEND_NAME), which means each row is uniquely defined by both the part and vendor.



PRODUCTSUPPLY	
PART_CODE	INTEGER
PART_DESC	VARCHAR(50)
VEND_NAME	VARCHAR(50)
VEND_ADDRESS	VARCHAR(100)
VEND_TYPE	VARCHAR(50)
PRICE	FLOAT

2) Remove Partial Dependencies (Normalization to 2NF)

The next step involves normalizing the table to **Second Normal Form (2NF)**, which means removing **partial dependencies**.

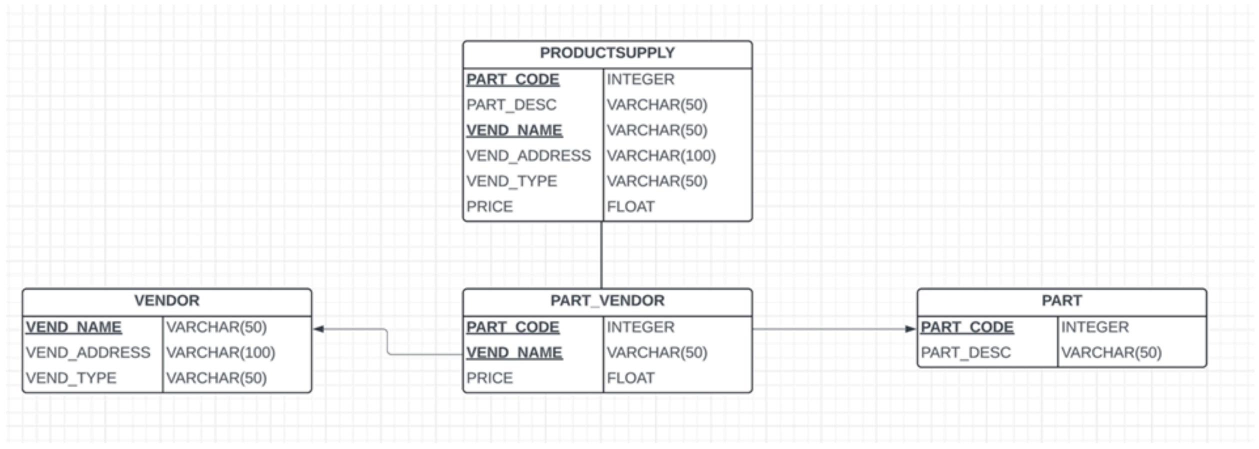
- A **partial dependency** occurs when a non-key attribute is only dependent on part of a composite primary key, instead of the whole composite key.
- In our case, the initial table has a composite key of PART_CODE and VEND_NAME, and attributes like PART_DESC and VEND_ADDRESS only depend on part of this composite key.

To remove partial dependencies, you need to create separate tables where non-key attributes are fully dependent on their primary keys.

Resulting Tables in 2NF:

1. **PART Table**:
   - **Attributes**: PART_CODE (Primary Key), PART_DESC
   - **Reason**: PART_DESC depends only on PART_CODE. This table stores unique parts and their descriptions.
2. **VENDOR Table**:
   - **Attributes**: VEND_NAME (Primary Key), VEND_ADDRESS, VEND_TYPE
   - **Reason**: VEND_ADDRESS and VEND_TYPE are fully dependent on VEND_NAME. This table stores vendor information.
3. **PART_VENDOR Table**:
   - **Attributes**: PART_CODE, VEND_NAME, PRICE
   - **Composite Primary Key**: (PART_CODE, VEND_NAME)
   - **Reason**: PRICE depends on both PART_CODE and VEND_NAME. This table stores information on which vendor supplies which part and at what price.

By creating these three tables, you have removed all **partial dependencies**. Each non-key attribute is fully dependent on its entire primary key.



3) Remove Transitive Dependencies (Normalization to 3NF)

The next step is to normalize the table further to **Third Normal Form (3NF)** by removing **transitive dependencies**.

- A **transitive dependency** occurs when a non-key attribute depends on another non-key attribute rather than directly on the primary key.
- After creating the `PART`, `VENDOR`, and `PART_VENDOR` tables, we need to ensure that no attribute depends on anything other than the primary key.

Evaluation of Transitive Dependencies:

1. **PART Table**:
   - **Primary Key**: `PART_CODE`
   - **Attributes**: `PART_DESC`
   - **Analysis**: `PART_DESC` depends directly on `PART_CODE`. There are no transitive dependencies here.
2. **`VENDOR` Table**:
   - **Primary Key**: `VEND_NAME`
   - **Attributes**: `VEND_ADDRESS`, `VEND_TYPE`

        - **Analysis**: `VEND_ADDRESS` and `VEND_TYPE` depend directly on `VEND_NAME`. There are no transitive dependencies here.
3. **`PART_VENDOR` Table**:
    - **Composite Primary Key**: (`PART_CODE`, `VEND_NAME`)
    - **Attribute**: `PRICE`
    - **Analysis**: `PRICE` depends directly on the composite key (`PART_CODE`, `VEND_NAME`). There are no transitive dependencies here.

Since there are no transitive dependencies in any of these tables, they are all in **3NF**.

4) **Step 4: Crow's Foot ERD**

Finally, draw the **Crow's Foot ERD** to illustrate the entities, attributes, relationships, primary keys, and foreign keys:

1. **Entities**:
    - **`PART`**:
        - **Attributes**: `id`, `part_code`, `part_desc`
        - **Primary Key**: `id`
    - **`VENDOR`**:
        - **Attributes**: `id`, `vend_name`, `vend_address`, `vend_type`
        - **Primary Key**: `id`
    - **`PART_VENDOR`**:
        - **Attributes**: `id`, `part_code` (Foreign Key referencing PART), `vend_name` (Foreign Key referencing `VENDOR`), `price`
        - **Primary Key**: `id`
2. **Relationships**:
    - **`PART` to `PART_VENDOR`**:
        - **One-to-Many** (`1 to *`): One part can be supplied by many vendors.

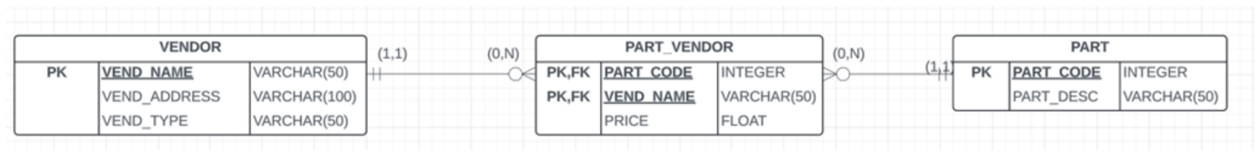
- **VENDOR to PART_VENDOR**:
    - **One-to-Many** (`1 to *`): One vendor can supply multiple parts.

The **Crow's Foot notation** should indicate the cardinality of relationships:

- A single **part** can have multiple records in the **PART_VENDOR** table.
- A single **vendor** can also have multiple records in the **PART_VENDOR** table.

The **Crow's Foot ERD** clearly identifies:

- **Primary Keys (PK)** and **Foreign Keys (FK)** for each table.
- Relationships between PART, VENDOR, and PART_VENDOR.


VENDOR
PK VEND_NAME VARCHAR(50)
VEND_ADDRESS VARCHAR(100)
VEND_TYPE VARCHAR(50)
(1,1)
(0,N)
PART_VENDOR
PK,FK PART_CODE INTEGER
PK,FK VEND_NAME VARCHAR(50)
PRICE FLOAT
(0,N)
(1,1)
PART
PK PART_CODE INTEGER
PART_DESC VARCHAR(50)