Database Management Systems

Pharmacy Supply Chain Management system

INDEX

INTRODUCTION

- A Pharmacy Supply Chain Management (SCM) system is indispensable for streamlining operations and ensuring efficiency across various entities within the pharmaceutical ecosystem.
- By leveraging SCM principles and technologies, pharmacies can effectively manage the flow of products and information from suppliers to customers, optimizing processes and enhancing overall performance.
- With SCM, pharmacies can maintain accurate inventory levels of medications and medical supplies, ensuring timely availability and minimizing stock outs or overstock situations.
- SCM enables pharmacies to establish seamless communication and collaboration with suppliers, facilitating efficient procurement processes and ensuring the timely delivery of products.
- SCM empowers pharmacies to enhance customer service by providing real-time visibility into order statuses and delivery schedules, thereby improving customer satisfaction and loyalty.
- By integrating SCM with transaction and order management systems, pharmacies can track the entire lifecycle of products, from procurement to dispensing, enabling better decision-making and resource allocation.

Overall, Pharmacy SCM systems play a pivotal role in optimizing supply chain operations, enhancing customer experiences, and driving overall business success in the pharmaceutical industry.

ASSUMPTIONS

- Initially, our model has 6 relations: Supplier, Pharmacy, Customer, Order, Product and Transaction.

- The relations **Supplier**, **Customer** and **Pharmacy** each contain the details of the participating entities in the pharmacy supply chain- namely a unique identifier for each, their name, phone number, pincode, area. Clearly, each pincode is associated with a unique area.

- The **Product** relation contains details about each product/item (medicines/other essentials) being sold or supplied- namely a unique identifier for each product, its name, dosage/specification of the medicine (mL/drops/other specs), category of the product (pill/capsule/other), manufacturing and expiry dates of the product, unit price, and most importantly, the supplier_id of the supplier who is supplying the product to the pharmacy, and the pharmacy_id of the pharmacy who is selling the product to the customer. Since multiple suppliers could supply the same product, and multiple pharmacies could also sell the same product, hence supplier_id and pharmacy_id are multivalued attributes. This relation also includes the qty_supplied by each supplier, and the qty_available to be sold in each pharmacy.
  Note that product_name and dosage have been considered as unique. Consider an example as follows: the "Pan" tablet has multiple different types with specifications, all manufactured by Alkem labs. When we consider the product_name alone, it is not unique. However, if we consider the product_name along with the dosage, this is unique (eg. Pan 40mg, Pan 80mg).

- The **Order** relation contains details about the order placed by customers/pharmacies. Obviously, either a customer can place an order to a pharmacy, or a pharmacy can place an order to a supplier. This relation

namely contains a unique identifier for each order, a list of all the product_ids of all the products which the order entails (multivalued attributes), the qty_ordered in each order, the order_date and delivery_date, the total price of the order, and the client_type. Client type specifies who is placing the order:

- ❖ Client_type = 'C': This means that a customer is placing an order to a pharmacy, who sells the product. In this case, the customer_id and pharmacy_id fields will be filled with the identifiers of the corresponding customer and pharmacy involved in the order, whereas the supplier_id field will be left NULL. (As no supplier is involved in this order.)
- ❖ Client_type = 'P': This means that a pharmacy is placing an order to a supplier, who supplies the product. In this case, the pharmacy_id and supplier_id fields will be filled with the identifiers of the corresponding pharmacy and supplier involved in the order, whereas the customer_id field will be left NULL. (As no customer is involved in this order.)

Note that at any given point in time, one of these 3 fields (supplier_id, pharmacy_id, customer_id) will be left NULL depending on whether the order is being placed by a customer to a pharmacy, or by a pharmacy to a supplier.

- The **Transaction** relation contains details about the transaction which has occurred between the participating entities for a particular order- namely a unique identifier for each transaction, the order_id of the order for which the transaction has occurred, the type of transaction (card/cash/UPI/other methods), the date of the transaction, and the status of the transaction:
  - ❖ Completed: The transaction has been completely paid off successfully.
  - ❖ In Progress: If the transaction is being paid in installments, then for each installment in the transaction, the state of the transaction will be marked "in progress".
  - ❖ Pending: The transaction has not begun yet.

Note that for a particular order, there could be multiple transactions taking place, since the client could choose to complete the payment for the order in installments instead of altogether. In this case, there will be multiple transaction_ids linked with each order_id.

ENTITIES AND ATTRIBUTES

1. Supplier (supplier_id, supplier_name, pincode, area, supplier_phno)
2. Pharmacy (pharmacy_id, pharmacy_name, pincode, area, pharmacy_phno)
3. Customer (customer_id, customer_name, pincode, area, customer_phno)
4. Product (product_id, product_name, dosage, product_category, mfg_date , exp_date, supplier_id, pharmacy_id, unit_price, qty_supplied, qty_available)
5. Order (order_id, product_id, qty_ordered, order_date, delivery_date, total_amount, client_type, customer_id, pharmacy_id,supplier_id)
6. Transaction (transaction_id, order_id, transaction_type, transaction_date, transaction_status)

FUNCTIONAL DEPENDENCIES

Step1: Listing the FDs:
(green = trivial)

It involves listing the functional dependencies (FDs) within the database schema, providing a clear understanding of the relationships between attributes and ensuring data integrity.

Supplier:

supplier_id -> supplier_name
supplier_id -> supplier_phno
supplier_id -> pincode, area
supplier_phno -> supplier_name
supplier_phno -> pincode, area
supplier_id, supplier_phno -> supplier_name
supplier_id, supplier_phno -> pincode, area
supplier_id, pincode-> area
pincode -> area
supplier_id, supplier_name -> supplier_name
supplier_id, supplier_phno -> supplier_phno

Pharmacy:

pharmacy_id -> pharmacy_name
pharmacy_id -> pharmacy_phno
pharmacy_id -> pincode, area
pharmacy_phno -> pharmacy_name
pharmacy_phno -> pincode, area
pharmacy_id, pharmacy_phno -> pharmacy_name
pharmacy_id, pharmacy_phno -> pincode, area
pharmacy_id, pincode-> area
pincode -> area
pharmacy_id, pharmacy_name -> pharmacy_name

pharmacy_id, pharmacy_phno -> pharmacy_phno

Customer :

customer_id -> customer_name
customer_id -> customer_phno
customer_id -> pincode, area
customer_phno -> customer_name
customer_phno -> pincode, area
customer_id, customer_phno -> customer_name
customer_id, customer_phno -> pincode, area
customer_id, pincode-> area
pincode -> area
customer_id, customer_name -> customer_name
customer_id, customer_phno -> customer_phno

Product:

product_id -> product_name
product_id -> dosage
product_id -> product_category
product_id -> supplier_id
product_id -> pharmacy_id
product_id -> exp_date
product_id -> mfg_date
product_id -> unit_price
product_id -> qty_available
product_id -> qty_supplied
product_name, dosage -> product_id
product_id, pharmacy_id -> qty_available
product_id, supplier_id -> qty_supplied
product_name, dosage -> product_category
product_name, dosage -> supplier_id
product_name, dosage -> pharmacy_id
product_name, dosage -> exp_date
product_name, dosage -> mfg_date
product_name, dosage -> unit_price

product_name, dosage -> qty_available
product_name, dosage -> qty_supplied
product_name-> product_category
product_id, product_name -> product_name
product_id, product_category -> product_category

Order:

order_id -> qty_ordered
order_id -> order_date
order_id -> delivery_date
order_id -> total_amount
order_id -> client_type
order_id -> product_id
order_id -> customer_id
order_id -> pharmacy_id
order_id -> supplier_id
order_id, product_id -> qty_ordered
order_id, product_id -> total_amount
order_id, delivery_date -> total_amount
order_id, order_date -> total_amount
order_id, customer_id -> client_type
order_id, pharmacy_id -> client_type
order_id, order_date -> order_date
order_id, customer_id -> customer_id

Transaction:

transaction_id -> order_id
transaction_id -> transaction_type
transaction_id -> transaction_date
transaction_id -> transaction_status
transaction_id, order_id -> transaction_date
transaction_id, order_id -> transaction_type
transaction_id, transaction_type -> order_id
transaction_id, transaction_type -> transaction_date

transaction_id, transaction_date -> transaction_type
transaction_id, transaction_date -> transaction_status
transaction_id, order_id -> order_id
transaction_id, transaction_type -> transaction_type

Step 2: Removing trivial and time dependent FDs
(green = trivial)

- Trivial functional dependencies (FDs) represent relationships where the determinant attributes uniquely determine the dependent attributes due to inherent constraints
- Time-dependent FDs capture dependencies that vary with time, reflecting changes in data over different temporal periods.

Supplier:
supplier_id -> supplier_name
supplier_id -> supplier_phno
supplier_id -> pincode, area
supplier_phno -> supplier_name
supplier_phno -> pincode, area
supplier_id, supplier_phno -> supplier_name
supplier_id, supplier_phno -> pincode, area
supplier_id, pincode-> area
pincode -> area
~~supplier_id, supplier_name -> supplier_name~~
~~supplier_id, supplier_phno -> supplier_phno~~

These dependencies are trivial as they state that the attribute "supplier_name" (respectively, "supplier_phno") is determined by itself along with the corresponding "supplier_id," which is inherent in their definitions.

Pharmacy:

pharmacy_id -> pharmacy_name
pharmacy_id -> pharmacy_phno
pharmacy_id -> pincode, area
pharmacy_phno -> pharmacy_name
pharmacy_phno -> pincode, area
pharmacy_id, pharmacy_phno -> pharmacy_name
pharmacy_id, pharmacy_phno -> pincode, area
pharmacy_id, pincode-> area
pincode -> area
~~pharmacy_id, pharmacy_name -> pharmacy_name~~
~~pharmacy_id, pharmacy_phno -> pharmacy_phno~~

These dependencies are trivial because they assert that the attribute "pharmacy_name" (and "pharmacy_phno") is determined by itself along with the corresponding "pharmacy_id," which is self-evident and inherent in their definitions.

Customer :

customer_id -> customer_name
customer_id -> customer_phno
customer_id -> pincode, area
customer_phno -> customer_name
customer_phno -> pincode, area
customer_id, customer_phno -> customer_name
customer_id, customer_phno -> pincode, area
customer_id, pincode-> area
pincode -> area
~~customer_id, customer_name -> customer_name~~
~~customer_id, customer_phno -> customer_phno~~

These dependencies are trivial as they state that the attribute "customer_name" (and "customer_phno") is determined by itself along with the corresponding "customer_id," which is inherent in their definitions.

Product :

product_id -> product_name
product_id -> dosage
product_id -> product_category
product_id -> supplier_id
product_id -> pharmacy_id
product_id -> exp_date
product_id -> mfg_date
product_id -> unit_price
product_id -> qty_available
product_id -> qty_supplied
product_name, dosage -> product_id
product_id, pharmacy_id -> qty_available
product_id, supplier_id -> qty_supplied
product_name, dosage -> product_category
product_name, dosage -> supplier_id
product_name, dosage -> pharmacy_id
product_name, dosage -> exp_date
product_name, dosage -> mfg_date
product_name, dosage -> unit_price
product_name, dosage -> qty_available
product_name, dosage -> qty_supplied
product_name-> product_category
~~product_id, product_name -> product_name~~
~~product_id, product_category -> product_category~~

These dependencies are trivial as they assert that the attribute "product_name" (and "product_category") is determined by itself along with the corresponding "product_id," which is inherent in their definitions.

Order:

order_id -> qty_ordered
order_id -> order_date
order_id -> delivery_date
order_id -> total_amount
order_id -> client_type
order_id -> product_id
order_id -> customer_id
order_id -> pharmacy_id
order_id -> supplier_id
order_id, product_id -> qty_ordered
order_id, product_id -> total_amount
order_id, delivery_date -> total_amount
order_id, order_date -> total_amount
order_id, customer_id -> client_type
order_id, pharmacy_id -> client_type
~~order_id, order_date -> order_date~~
~~order_id, customer_id -> customer_id~~

These dependencies are trivial as they state that the attribute "order_date" (and "customer_id") is determined by itself along with the corresponding "order_id," which is inherent in their definitions.

Transaction:

transaction_id -> order_id
transaction_id -> transaction_type
transaction_id -> transaction_date
transaction_id -> transaction_status
transaction_id, order_id -> transaction_date
transaction_id, order_id -> transaction_type
transaction_id, transaction_type -> order_id
transaction_id, transaction_type -> transaction_date
transaction_id, transaction_date -> transaction_type
transaction_id, transaction_date -> transaction_status

~~transaction_id, transaction_type -> transaction_type~~
~~transaction_id, order_id -> order_id~~

These dependencies are trivial as they assert that the attribute "transaction_type" (and "order_id") is determined by itself along with the corresponding "transaction_id," which is inherent in their definitions.

MINIMAL CLOSURE/CANONICAL COVER OF FDs

① Supplier Table

Attributes : supplier_id (A), supplier_name (B), pincode (C), area (D), supplier_phono (E)

FDs

$A \to B$
$A \to E$
$A \to C, D$
$E \to B$
$E \to C, D$
$A, E \to B$
$A, E \to C, D$
$A, C \to D$
$C \to D$

Step 1 : Making all RHS of FDs as Single-ton.

∴ Decomposing,

$A \to C, D$: $A \to C$, $A \to D$

$A, E \to C, D$: $A, E \to C$, $A, E \to D$

$E \to C, D$: $E \to C$, $E \to D$

Step 2 : Checking for extraneous

(i) $A, E \to B$

If E is extraneous, removing E

$A \to B$: $\{A^+\} = \{A, B, C, D, E\}$ ∴ E is extraneous

If A is extraneous, removing A

$E \to B$: $\{E^+\} = \{E, B, C, D\}$ Doesn't include A

∴ A is not extraneous.

∴ FD becomes $A \to B$

(ii) $A, E \to C$

If E is extraneous, removing E

$A \to C$: $\{A^+\} = \{A, B, E, C, D\}$ ∴ E is extraneous

If A is extraneous, removing A

$E \to C$: $\{E^+\} = \{E, B, C, D\}$ Doesn't include A

∴ A is not extraneous

∴ FD becomes $A \to C$

(iii) $A, E \to D$

If E is extraneous, removing E

$A \to D$ $\{A^+\} = \{A, B, E, C, D\}$ ∴ E is extraneous

Checking by removing A

$E \to D$: $\{E^+\} = \{E, B, C, D\}$ Doesn't include A

∴ A is not extraneous

∴ FD becomes $A \rightarrow D$

v) $A, C \rightarrow D$

Similarly, we can deduce that C is extraneous & A is not

∴ FD becomes $A \rightarrow D$

Step 3 : Checking for Redundant FDs.

for $A \rightarrow B$, removing $A \rightarrow B$

$\{A^+\} = \{A, E, C, D, B\}$ ∴ $A \rightarrow B$ is redundant as we obtain closure

for $E \rightarrow B$, removing $E \rightarrow B$

$\{E^+\} = \{E, C, D\}$ Doesn't contain all attributes

∴ $E \rightarrow B$ is not redundant.

Following this step for all the reduced FDs in step 2, we obtain minimal closure as

$A \rightarrow E$	Supplier_id → phno
$E \rightarrow B$	Supplier_phno → name
$E \rightarrow C$	Supplier_phone → pincode
$C \rightarrow D$	pincode → area

② Pharmacy Table

Attributes: pharmacy-id, pharmacy-name, pincode, area, pharmacy-phno
A B C D E

FDs
$A \rightarrow B$
$A \rightarrow E$
$A \rightarrow C, D$
$E \rightarrow B$
$E \rightarrow C, D$
$A, E \rightarrow B$
$A, E \rightarrow C, D$
$A, C \rightarrow D$
$C \rightarrow D$

Step 1: Making all RHS of FDs into Single-ton Decomposing

$A \rightarrow C, D$: $A \rightarrow C$, $A \rightarrow D$

$A, E \rightarrow C, D$: $A, E \rightarrow C$, $A, E \rightarrow D$

$E \rightarrow C, D$: $E \rightarrow C$, $E \rightarrow D$

Step 2: Checking for extraneous

(i) $A, C \rightarrow D$

If C is extraneous, removing C

$A \rightarrow D$: $\{A^*\} = \{A, B, E, C, D\}$ $\therefore$ C is extraneous

If A is extraneous, removing A

$C \rightarrow D$: $\{C^*\} = \{C, D\}$ Doesn't include A

$\therefore$ A is not extraneous.

$\therefore$ FD becomes $A \rightarrow D$

(ii) $A, E \rightarrow B$ (iii) $A, E \rightarrow C$ (iv) $A, E \rightarrow D$

Similarly, we can deduce that E is extraneous & A is not

$\therefore$ FDs become $A \rightarrow B$
$A \rightarrow C$
$A \rightarrow D$

Step 3: Checking for Redundant FDs.

For $A \rightarrow D$, removing $A \rightarrow D$

$\{A^*\} = \{A, B, E, C, D\}$ $\therefore$ $A \rightarrow D$ is redundant as we obtain closure

For $C \rightarrow D$, removing $C \rightarrow D$

$\{C^*\} = \{C, D\}$ Doesn't contain all attributes

$\therefore$ $C \rightarrow D$ not redundant

Following above step for all the reduced FDs in step 3, we obtain minimal closure as

$A \rightarrow E$	Pharm_id → Phno
$E \rightarrow B$	Pharm_pho → pharm_name
$E \rightarrow C$	Pharm_phno → pincode
$C \rightarrow D$	pincode → area

③ Customers Table

Attributes : customer_id, customer_name, pincode, area, pharmacy_phno
A B C D E

FDs
$A \to B$
$A \to E$
$A \to C, D$
$E \to B$
$E \to C, D$
$A, E \to B$
$A, E \to C, D$
$A, C \to D$
$C \to D$

Step 1: Making all the RHS of FDs into Singleton
Decomposing

$A \to C, D$: $A \to C$, $A \to D$
$A, E \to C, D$: $A, E \to C$, $A, E \to D$
$E \to C, D$: $E \to C$, $E \to D$

Step2 : Checking for extraneous

(i) $A, C \to D$
If C is extraneous removing C.
$A \to D$: $\{A^+\} = \{A, B, E, C, D\}$ ∴ C is extraneous.
If A is extraneous, removing A
$C \to D$: $\{C^+\} = \{C, D\}$ Doesn't include A
∴ A is not extraneous.
∴ FD becomes $A \to D$

(ii) $A, E \to B$ (iii) $A, E \to C$ (iv) $A, E \to D$
Similarly, we can deduce that E is extraneous & A is not
∴ FDs become $A \to B$
$A \to C$
$A \to D$

Step 3 : Checking for Redundant FDs
For $A \to D$, removing $A \to D$
$\{A^+\} = \{A, B, E, C, D\}$ ∴ $A \to D$ is redudant as we obtain closure
For $C \to D$ removing $C \to D$
$\{C^+\} = \{C, D\}$ Doesn't contain all attributes
∴ $C \to D$ not redundant.

Following above step for all the reduced FDs in step 3, we obtain minimal closure as

$A \to E$	Cust_id → phno
$E \to B$	cust_phno → cust_name
$E \to C$	cust_phno → pincode
$C \to D$	pincode → area

④ Orders Table

Attributes : Order_id (A), product_id (B), qty_ordered (C), order_date (D), delv_date (E), total_amount (F), client_type (G), customer_id (H), pharmacy_id (I), supplier_id (J)

FDs

$A \to C$	$A \to G$	$A \to J$	$A, D \to F$
$A \to D$	$A \to B$	$A, B \to C$	$A, H \to G$
$A \to E$	$A \to H$	$A, B \to F$	$A, I \to G$
$A \to F$	$A \to I$	$A, E \to F$	

Step 1 : All elements on the RHS are singleton

Step2 : Checking for extraneous

(i) $A, B \to C$

If B is extraneous, removing B

$A \to C$: $\{A^+\} = \{A, C, D, E, F, G, H, B, I\}$ ∴ B is redundant

If A is extraneous, removing A

$B \to C$: $\{B^+\} = \{B, C\}$ Doesn't include A

∴ A is not extraneous

∴ FD becomes $A \to C$

(ii) $A, B \to F$

IIIrly, we can deduce that B is extraneous & A is not

∴ FD becomes $A \to F$

(iii) $A, E \to F$

IIIrly, we can deduce that E is extraneous & A is not

∴ FD becomes $A \to F$

(iv) $A, D \to F$

IIIrly, we can deduce that D is extraneous & A is not

∴ FD becomes $A \to F$

(v) $A, H \to G$

IIIrly, we can deduce that H is extraneous & A is not

∴ FD becomes $A \to G$

(vi) $A, I \to G$

IIIrly, we can deduce that I is extraneous & A is not

∴ FD becomes $A \to G$.

Step3 : Checking for Redundant FDs

For $A \to C$ removing $A \to C$

$\{A^+\} = \{A, B, D, E, F, G, H, I\}$ C is not included

∴ $A \to C$ is not redundant

Following above step for all the reduced FDs in step 3, we obtain minimal closure as

A → D	Order_id → ord_date
A → E	ord_id → delv_date
A → B	ord_id → product_id
A → H	ord_id → cust_id
A → I	ord_id → pharm_id
A → J	ord_id → supp_id
A → C	ord_id → qty_ord
A → F	order_id → total_amt
A → G	order_id → client_type

⑤ Products Table

Attributes : Product_id (A), Product_name (B), dosage (C), product_category (D), mfg-date (E), exp-date (F), supplier_id (G), pharmacy_id (H), unit_price (I), qty-supplied (J), qty-available (K)

FDs

A → B	A → F	A, H → K	B, C → H	B, C → J
A → C	A → E	A, G → J	B, C → F	B → D
A → D	A → I	B, C → A	B, C → E	
A → G	A → K	B, C → D	B, C → I	
A → H	A → J	B, C → G	B, C → K	

Step 1 : All elements on the RHS are Singleton

Step 2 : Checking for extraneous

(i) A, H → K

If H is extraneous, removing H

A → K : {A⁺} = {A, B, C, D, G, H, F, E, I, K, J} ∴ H is redundant

If A is extraneous, removing A

H → K : {H⁺} = {H, K} Doesn't include A

∴ A is not extraneous

∴ FD becomes A → K.

Similarly for the other FDs with multiple attributes on the LHS of the FDs gets simplified & redandant/extraneous attributes are removed.

Step 3 : Checking for redundant FDs

For A → G, removing A → G

{A⁺} = {A, B, C, D, E, F, G, H, I, J, K}

∴ A → G is redundant as we obtain closure

For A → B, removing A → B

{A⁺} = {A, C, D, E, F, G, H, I, J, K} B is not included

∴ A → B is not redundant

Following above step for all the reduced FDs in step 3, we obtain minimal closure as

A → B	Product_id → prod_name
A → C	prod_id → dosage
B, C → A	prod_name, dosage → prod_id
B, C → G	prod_name, dosage → supp_id
B, C → H	prod_name, dosage → pharm_id
B, C → F	prod_name, dosage → exp_date
B, C → E	prod_name, dosage → manf_date
B, C → I	prod_name, dosage → unit_price
B, C → K	prod_name, dosage → qty_avail
B, C → J	prod_name, dosage → qty_supplier
B → D	prod_name → prod_category

⑥ Transaction Table

Attributes: transaction_id (A), order_id (B), transaction_type (C), transaction_date (D), transaction_status (E)

FDs

$A \to B$	$A, B \to D$	$A, D \to C$
$A \to C$	$A, B \to C$	$A, D \to E$
$A \to D$	$A, C \to B$	
$A \to E$	$A, C \to D$	

Step 1: All the elements on the RHS are singleton

Step 2: Checking for extraneous

(i) $A, B \to D$

If B is extraneous, removing B

$A \to D$: $\{A^+\} = \{A, B, C, D, E\}$ ∴ B is extraneous

If A is extraneous, removing A

$B \to D$: $\{B^+\} = \{B, D\}$ Doesn't include A

∴ A is not extraneous

FD becomes $A \to D$

(ii) $A, B \to C$

Similarly, we can deduce that B is extraneous & A is not

∴ FD becomes $A \to C$

(iii) $A, C \to B$ (iv) $A, C \to D$

Similarly, we can deduce that C is extraneous & A is not

∴ FDs becomes $A \to B$, $A \to D$

(v) $A, D \to C$ (vi) $A, D \to E$

Similarly, we can deduce that D is extraneous & A is not

∴ FDs become $A \to C$, $A \to E$

Step3 : Checking for redundant

For $A \rightarrow B$, removing $A \rightarrow B$

$\{A^+\} = \{A, C, D, E\}$ B is not included

$\therefore A \rightarrow B$ is not redundant

For $A \rightarrow C$, removing $A \rightarrow C$

$\{A^+\} = \{A, B, D, E\}$ C is not included

$\therefore A \rightarrow C$ is not redundant

Following above step for all the reduced FDs in step 3, we obtain the following minimal closure is obtained :

$A \rightarrow B$	transac_id → order_id
$A \rightarrow C$	transac_id → transac_type
$A \rightarrow D$	transac_id → transac_date
$A \rightarrow E$	transac_id → transac_status

IDENTIFYING THE CANDIDATE KEY

Supplier:

Entity -: Supplier

attributes -: Supplier_id (A), supplier_phno (B), Supplier_name (C)

Pincode (D), Area (E)

Minimal cover -: $\{A \to B, B \to C, B \to D, D \to E\}$

Now

$\{A^+\} = \{A, B, C, D, E\}$ => Contains all the attributes

$\{B^+\} = \{B, C, D, E\}$

$\{D^+\} = \{D, E\}$

} => These two does not contain all the attributes; even it coming together.

=> Candidate key = A

Primary key is A i.e. supplier_id

Customer:

Entity : Customer

attributes : Customer_id , Customer_phno, Customer_name
(A) (B) (C)

Pincode , Area
(D) (E)

Minimal cover -: $\{A \to B, B \to C, B \to D, D \to E\}$

Now,

$\{A^+\} = \{A, B, C, D, E\}$ ⇒ This includes all the attributes.

$\{B^+\} = \{B, C, D, E\}$
$\{D^+\} = \{D, E\}$
⇒ These two do not have all the attributes

⇒ Candidate Key = A

⇒ Primary Key = A i.e. Customer_id.

Pharmacy:

Entity : Pharmacy

attributes -: pharmacy-id (A), pharmacy-phno (B), pharmacy-name (C)

Pincode (D), Area (E)

Minimal cover -: $\{A \to B, B \to C, B \to D, D \to E\}$

Now,

$\{A^+\} = \{A, B, C, D, E\}$ => This includes all the attributes

$\{B^+\} = \{B, C, D, E\}$

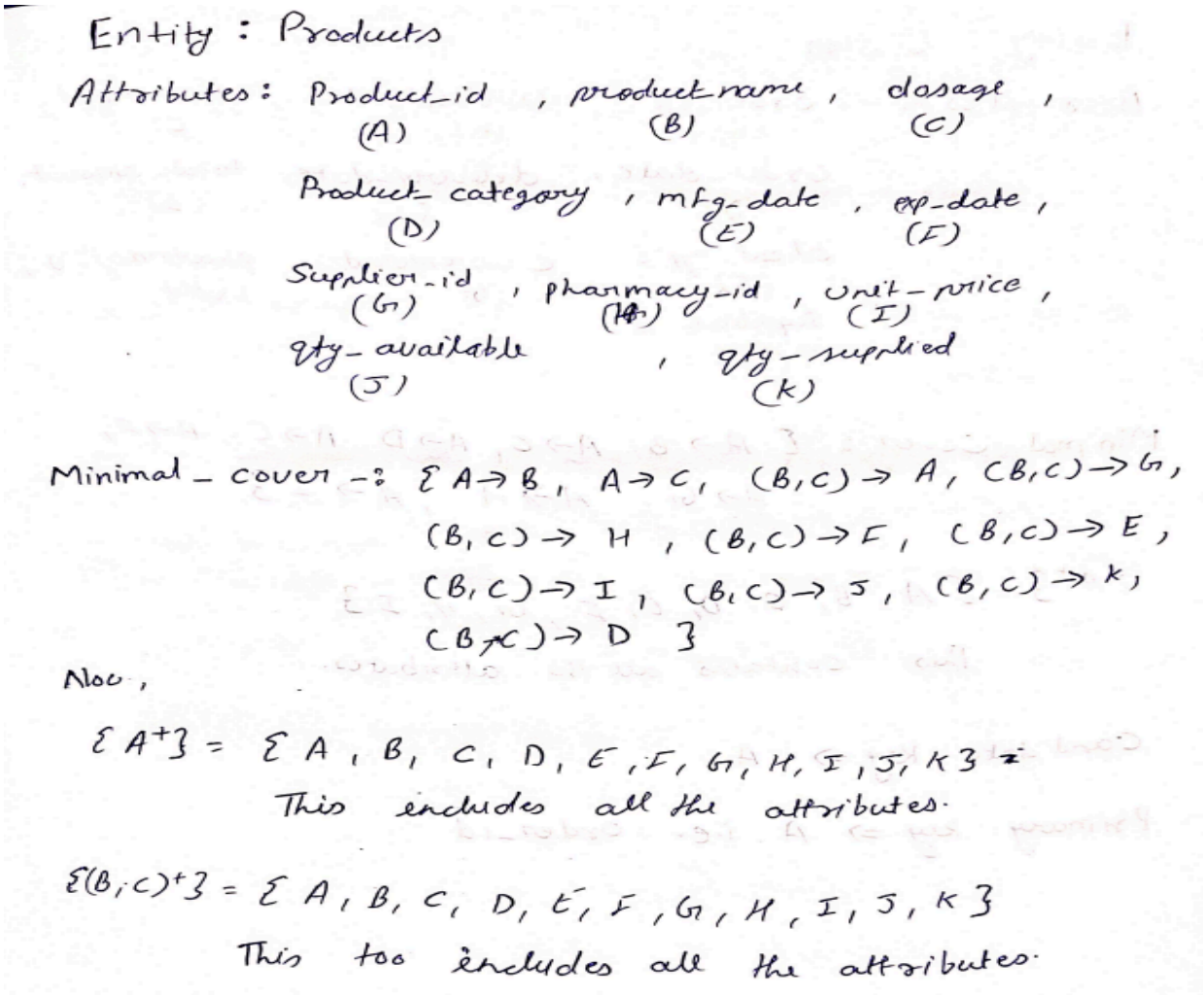
$\{D^+\} = \{D, E\}$

=> These two do not have all the attributes.

=> Candidate Key = A

=> Primary Key = A i.e. pharmacy-id

Products:



Candidate Keys: Product_ID
{Product_name, dosage}

Primary Key: Product_ID

We choose Product_ID as the primary key over {Product_name, dosage}, as product_id is likely to be unique and stable, not changing over time, whereas product_name and dosage might change. For example, product names might be updated for branding purposes, and dosages could be adjusted due to regulatory changes. A single-column primary key also minimizes the risk of errors during data entry and updates, as there is only one field to manage.

Order:

Entity : Order

Attributes -: order_id (A), product_id (B), qty_ordered (C), order_date (D), delivery_date (E), total_amount (E), client_type (F), customer_id (G), pharmacy_id (H), supplier_id (I)

Minimal Cover : $\{ A \to B, A \to C, A \to D, A \to E, A \to F, A \to G, A \to H, A \to I \}$

$\{A^+\} = \{ A, B, C, D, E, I, G, H, I \}$

This contains all the attributes.

Candidate Key => A

Primary Key => A i.e. order_id

Transaction:

Entity : Transaction

Attributes : transaction_id (A), order_id (B), transaction_date (C)

transaction_type (D), transaction_status (E)

Now,

Minimal cover :

$\{ A \rightarrow B, A \rightarrow C, A \rightarrow D, A \rightarrow E \}$

$\{A^+\} = \{A, B, C, D, E\}$ => contains all the attributes.

Candidate Key => A

Primary Key => A i.e. transaction_id

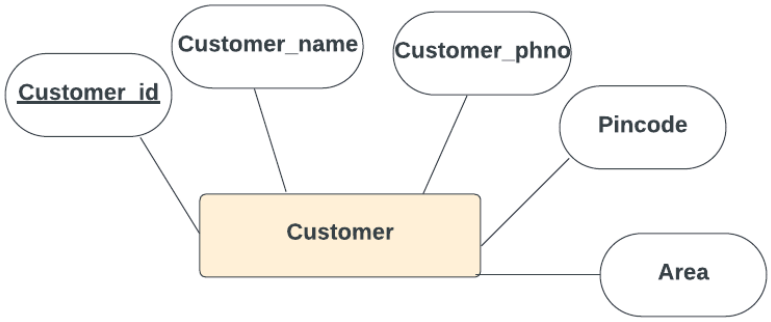
ENTITIES AND ATTRIBUTES WITH PK & FK

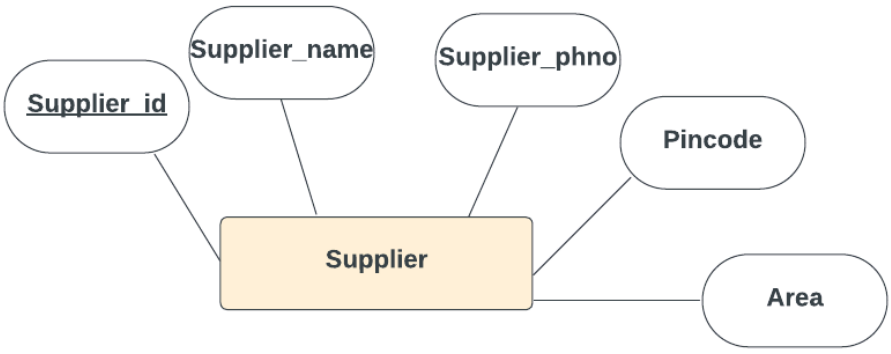
1. Supplier (**supplier_id**, supplier_name, pincode, area, supplier_phno)
2. Pharmacy (**pharmacy_id**, pharmacy_name, pincode, area, pharmacy_phno)
3. Customer (**customer_id**, customer_name, pincode, area, customer_phno)
4. Product (**product_id**, product_name, dosage, product_category, mfg_date , exp_date, supplier_id, pharmacy_id, unit_price, qty_supplied, qty_available)
5. Order (**order_id**, product_id, qty_ordered, order_date, delivery_date, total_amount, client_type, customer_id, pharmacy_id, supplier_id)
6. Transaction (**transaction_id**, order_id, transaction_type, transaction_date, transaction_status)

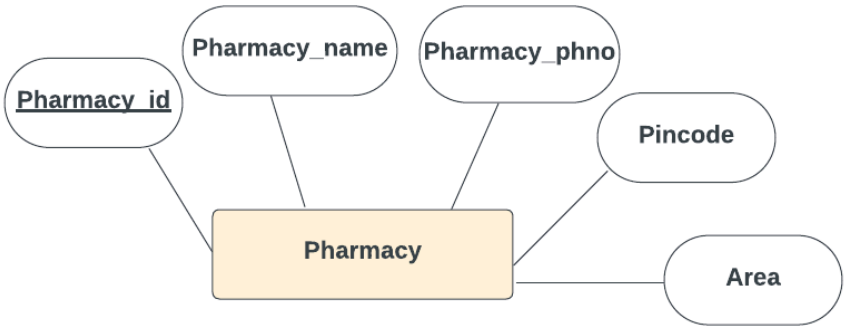
ENTITY RELATIONSHIP DIAGRAM

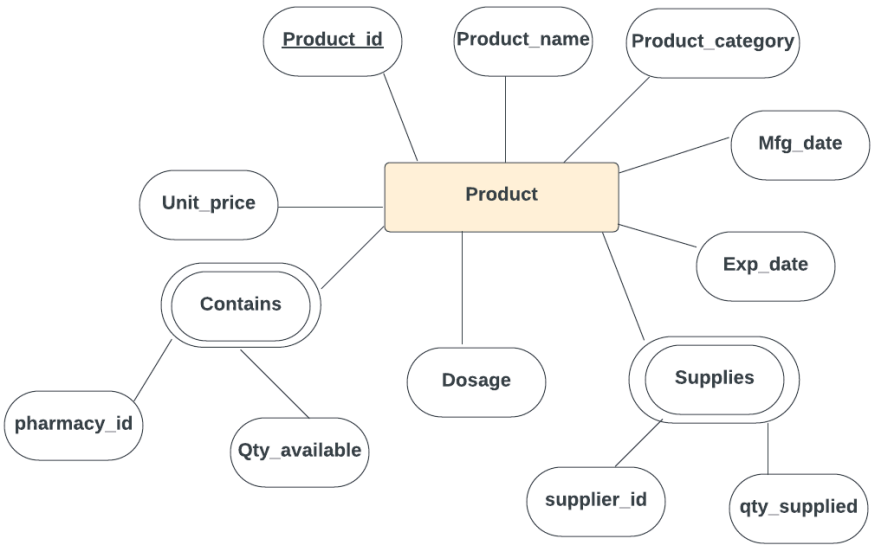
Step 1:

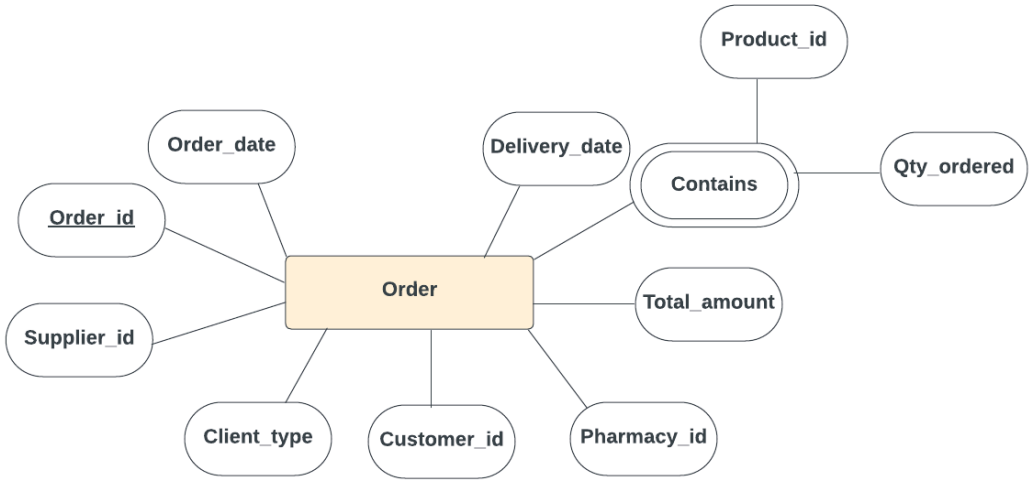
- The diagram is carefully constructed by identifying entities and defining their attributes while ensuring they adhere to predefined rules and constraints governing their relationships and integrity.
- If the attribute is multi-valued , it is represented as a double oval like in the case of Product_ID in the entity Order. The possible Primary Keys are underlined.

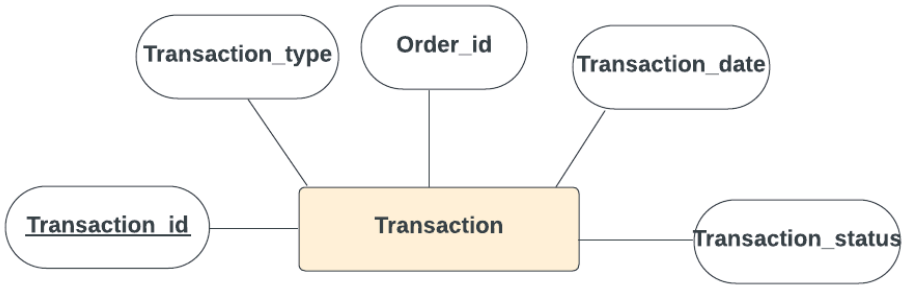





Pharmacy_name
Pharmacy_phno
Pharmacy_id
Pincode
Pharmacy
Area

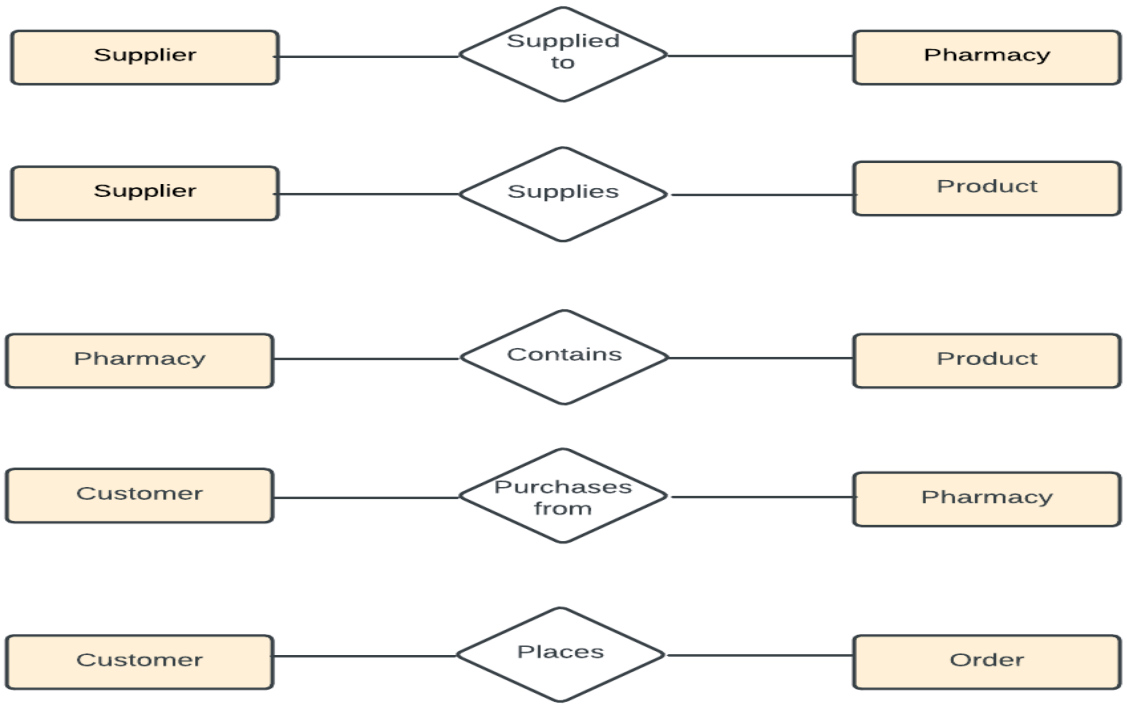

Product_id
Product_name
Product_category
Mfg_date
Unit_price
Product
Exp_date
Contains
Dosage
Supplies
pharmacy_id
Qty_available
supplier_id
qty_supplied

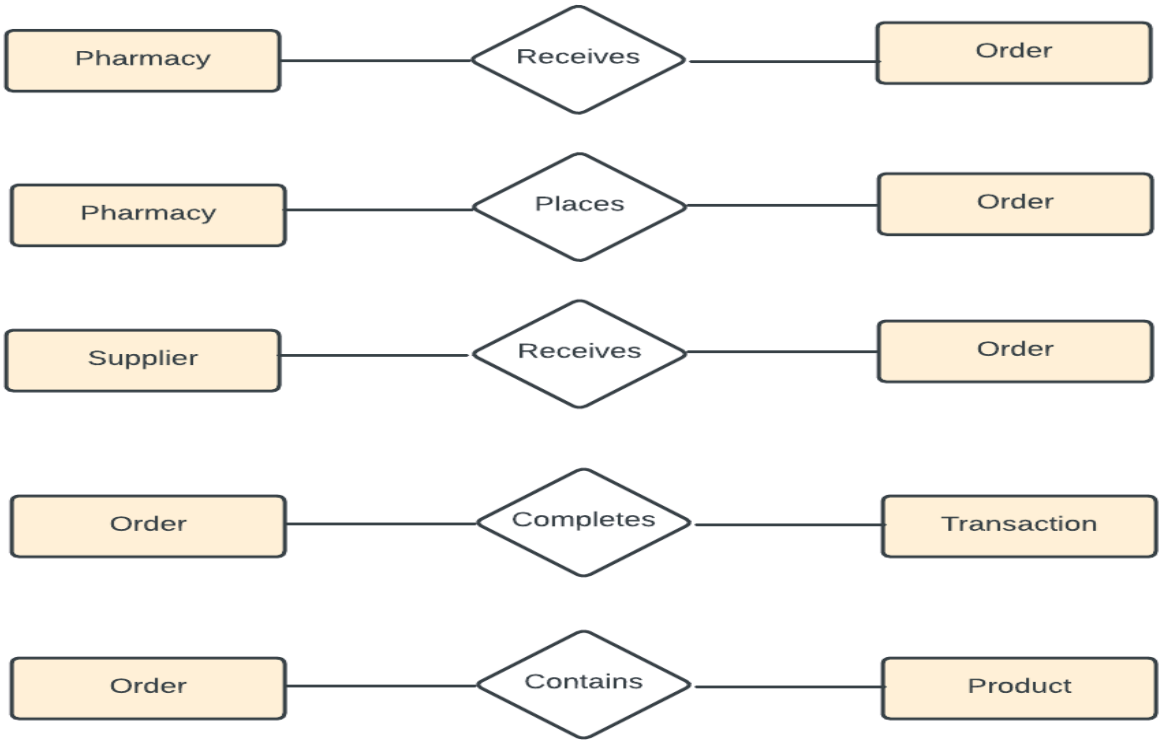

Product_id
Order_date
Delivery_date
Contains
Qty_ordered
Order_id
Order
Total_amount
Supplier_id
Client_type
Customer_id
Pharmacy_id


Transaction_type
Order_id
Transaction_date
Transaction_id
Transaction
Transaction_status

Step 2:

- It involves identifying the relationships between the entities, analyzing their connections and interactions within the system to develop a comprehensive understanding of the data model's structure and functionality.

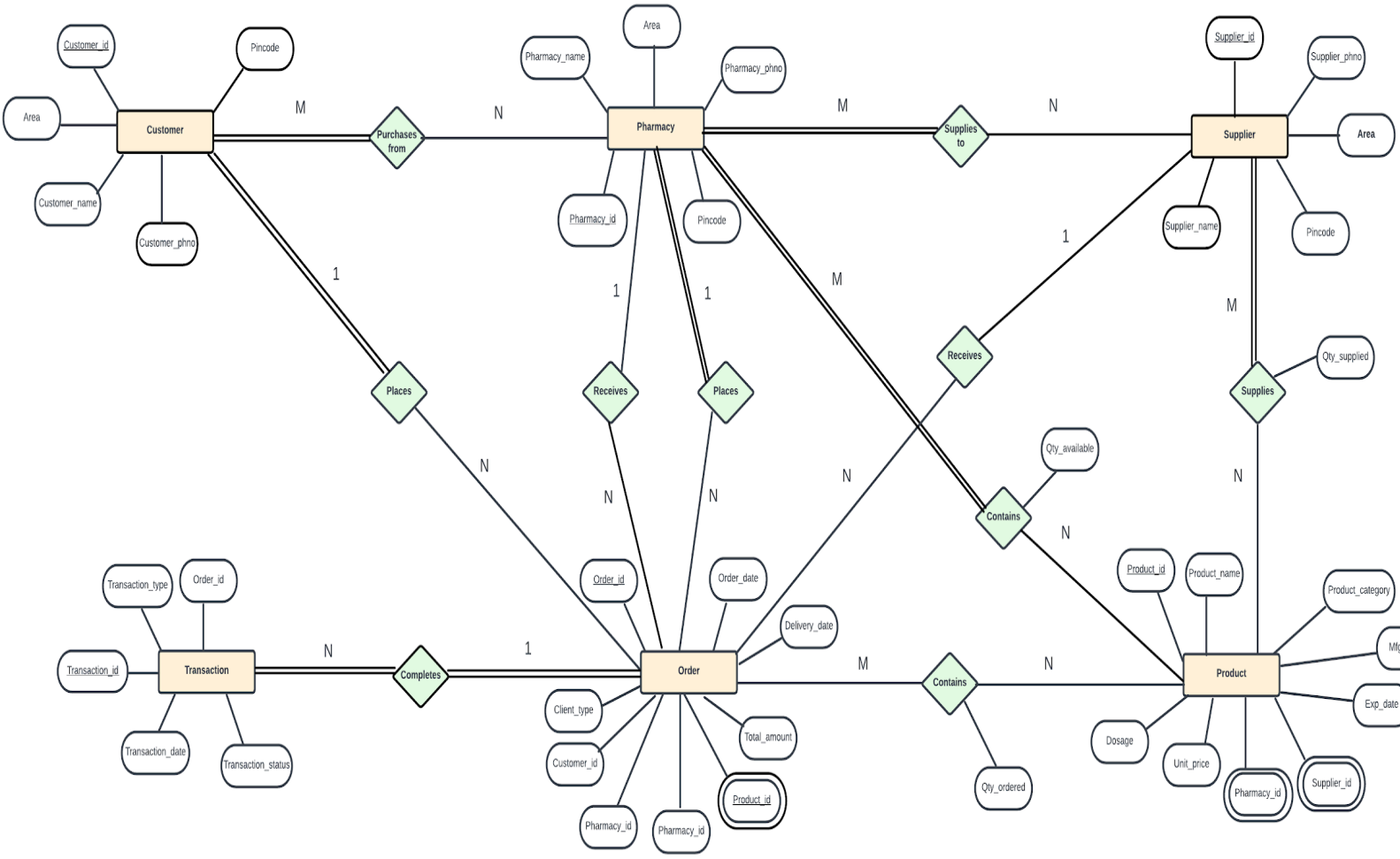




Step 3:

- Step 3 entails combining the outcomes of steps 1 and 2, producing a final Entity-Relationship (ER) diagram that integrates both entities and attributes along with their relationships. This process involves translating the identified entities and their attributes into a visual representation that illustrates their interconnectedness and dependencies within the system.

ER Diagram:



Link for reference:
https://lucid.app/lucidchart/60d66ca1-6565-4bf4-81ad-cb6d4fe8fdb8/edit?viewport_loc=-10%2C-10%2C1707%2C789%2C0_0&invitationId=inv_83cbf056-d050-4d05-b6ad-cd6ff64a38f3

SCHEMA DIAGRAM

CONVERTING ER DIAGRAM TO SCHEMA DIAGRAM:

Representing entities:

- Since all the entities present in our model are strong entity types, a relation, or table, is directly created for each.

Representing relationships:

a) 1:1 relationship/cardinality (one-to-one):
   - When 2 entities are related by a one-to-one cardinality, the primary key of one of the entities is linked as a foreign key to the other, or they share the same primary key.

b) 1:N relationship/cardinality (one-to-many):
   - When 2 entities are related by a one-to-many cardinality, the primary key of the "one" entity is linked as a foreign key to the "many" entity.
   - This is illustrated in the Customer-places-Order or Supplier-receives-Order relationship in our ER.

c) N:1 relationship/cardinality (many-to-one):
   - When 2 entities are related by a one-to-many cardinality, the primary key of the "one" entity is linked as a foreign key to the "many" entity
   - This is illustrated in the Order-completes-Transaction relationship in our ER.

d) M:N relationship/cardinality (many-to-many):
   - When 2 entities are related by a many-to-many cardinality, the primary key of one entity is linked as a foreign key to the other entity.
   - Alternatively, a junction table could be used.
   - This is illustrated in the Supplier-supplies to-Pharmacy, Supplier-supplies to-Pharmacy or Pharmacy-contains-Product

relationship in our ER.

Representing attributes:

a) Primary Key:
- The primary key of each table is underlined in the Schema diagram.

b) Foreign key:
- The foreign key of each table is linked to the corresponding primary key of the table, which it references.

c) Composite attributes:
- If an attribute in an entity can be decomposed into 2 or more simple attributes, that composite attribute is ignored, and the individual simple attributes which it can be broken down into are considered in the Schema diagram.

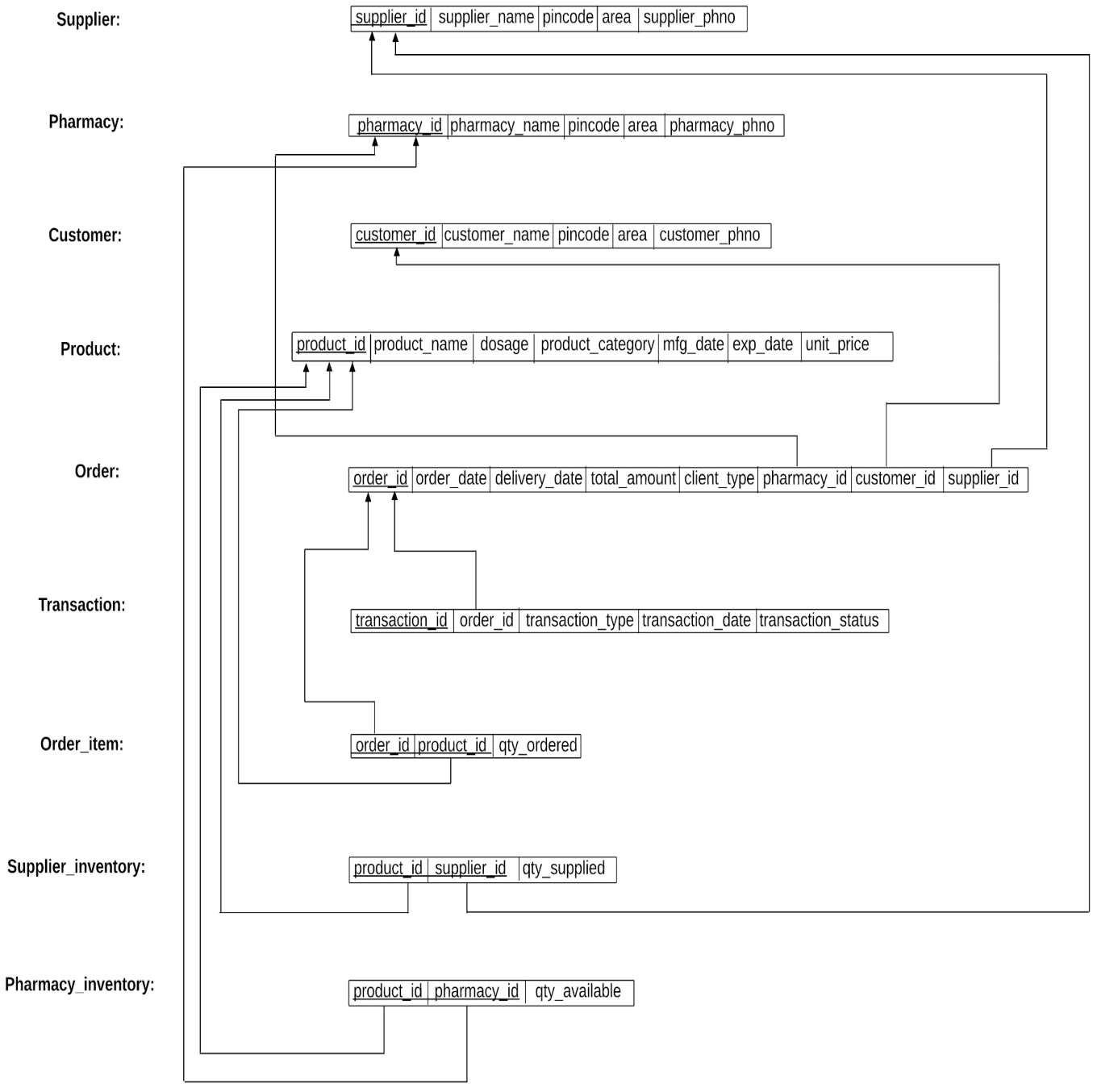
d) Multivalued attributes:
- If an attribute in an entity can take multiple values, an additional relation/entity has to be taken into account in the Schema.
- This additional relation/entity contains the multivalued attribute, along with the primary key of the entity, together as a **composite key**.
- These 2 attributes which function as the composite key in this additional relation/entity also act as foreign keys, linking to the corresponding primary keys of the parent relations.
- This is illustrated in the Order entity in our ER.
- In the entity Order, product_id is a multivalued attribute (since a single order can entail multiple products).
- In order to resolve this in the Schema diagram, an additional entity Order_Item has been introduced
- This new entity Order_Item contains the multivalued attribute product_id, along with the primary key order_id of the Order table. Together, (product_id, order_id) forms a composite key.
- Hence, each order can be related to a specific quantity of multiple products.
- Similarly, in the entity Product, pharmacy_id and supplier_id are

multivalued attributes (since a single product can exist in multiple pharmacies and a product can be supplied by multiple suppliers).

- In order to resolve this in the Schema diagram, additional entities Pharmacy_Inventory and Supplier_Inventory have been introduced

All other attributes (simple, single, non-key, etc.) will remain as it is in the Schema.

Schema Diagram:



REVISED SET OF ENTITIES AND ATTRIBUTES INVOLVED

1. Relation name-Supplier

Attribute	Purpose	Attribute type	Constraint
Supplier_ID	ID of the Supplier- S001	VARCHAR2(4)	PRIMARY KEY
Supplier_name	Name of the Supplier	VARCHAR2(50)	
Pincode	Pincode of the city	NUMBER(6)	FOREIGN KEY(From Location)
Area	Area in which the supplier operates from	VARCHAR2(40)	
Supplier_phno	Phone number of the Supplier	NUMBER(10)	

2. Relation name- Pharmacy

Attribute	Purpose	Attribute type	Constraint
Pharmacy_ID	ID of the Pharmacy- PH001	VARCHAR2(5)	PRIMARY KEY
Pharmacy_name	Name of the	VARCHAR2(50)	

	Pharmacy		
Pincode	Pincode of the city	NUMBER(6)	FOREIGN KEY(From Location)
Area	Area in which the pharmacy is located	VARCHAR2(40)	
Pharmacy_phno	Phone number of the Pharmacy	NUMBER(10)	

3. Relation name- Customer

Attribute	**Purpose**	**Attribute type**	**Constraint**
Customer_ID	ID of the Customer- C001	VARCHAR2(4)	PRIMARY KEY
Customer_name	Name of the Customer	VARCHAR2(50)	
Pincode	Pincode of the city	NUMBER(6)	FOREIGN KEY(From Location)
Area	Area in which the customer is located	VARCHAR2(40)	
Customer_phno	Phone number of the Customer	NUMBER(10)	

4. Relation name- Product

Attribute	**Purpose**	**Attribute type**	**Constraint**
Product_ID	ID of the Product- P001	VARCHAR2(4)	PRIMARY KEY
Product_name	Name of the Product	VARCHAR2(50)	
Product_category	Category the product belongs to - Tablet, Syrup….	VARCHAR2(50)	
Mfg_date	Manufacturing date of the product	DATE	
Exp_date	Expiry date of the product	DATE	
Unit_price	Unit price of each product	NUMBER(5)	
Dosage	Grammage or dosage specification od	VARCHAR2(50)	

5. Relation name-Pharmacy_Inventory

Attribute	**Purpose**	**Attribute type**	**Constraint**
Pharmacy_ID	ID of the Pharmacy- PH001	VARCHAR2(5)	COMPOSITE PRIMARY KEY, FOREIGN KEY(From Pharmacy)
Qty_available	Quantity available of each product	NUMBER(5)	
Product_ID	ID of the Product-P001	VARCHAR2(4)	COMPOSITE PRIMARY KEY, FOREIGN KEY(From products)

6. Relation name-Supplier_Inventory

Attribute	**Purpose**	**Attribute type**	**Constraint**
Supplier_ID	ID of the Supplier-S001	VARCHAR2(4)	COMPOSITE PRIMARY KEY, FOREIGN KEY(From Supplier)
Qty_supplied	Quantity supplied of each product	NUMBER(5)	
Product_ID	ID of the Product-P001	VARCHAR2(4)	COMPOSITE PRIMARY KEY, FOREIGN KEY(From products)

7. Relation name-Order

Attribute	**Purpose**	**Attribute type**	**Constraint**
Order_ID	ID of the Order- O001	VARCHAR2(4)	PRIMARY KEY
Order_date	Date on which the order is placed	DATE	
delivery_date	Date on which the order is delivered	DATE	
Total_amount	Total amount of the order placed	NUMBER(6)	
client_type	Type of client placing the order- Customer/ Pharmacy	VARCHAR2(50)	
Customer_ID	ID of the Customer- C001	VARCHAR2(4)	FOREIGN KEY(From Customer)
Pharmacy_ID	ID of the Pharmacy- PH001	VARCHAR2(5)	FOREIGN KEY(From Pharmacy)
Supplier_ID	ID of the Supplier- S001	VARCHAR2(4)	FOREIGN KEY(From Supplier)

8. Relation name-Transaction

Attribute	**Purpose**	**Attribute type**	**Constraint**
Transaction_ID	ID of the Transaction- T001	VARCHAR2(4)	PRIMARY KEY
Transaction_ status	Transaction status - Full/ Partial Payment, Refund request	VARCHAR2(50)	
Transaction_date	Date on which the transaction is made	DATE	
Transaction_type	Mode of payment - Cash,Card,net…	VARCHAR2(50)	
Order_ID	ID of the Order-O001	VARCHAR2(4)	FOREIGN KEY(From Order)

9. Relation name-Order_Item

Attribute	**Purpose**	**Attribute type**	**Constraint**
Order_ID	ID of the Order-O001	VARCHAR2(4)	FOREIGN KEY(From Order), COMPOSITE PRIMARY KEY
Product_ID	ID of the Product-P001	VARCHAR2(50)	FOREIGN KEY(From Product), COMPOSITE PRIMARY KEY
Quantity_ordered	Quantity of product ordered	NUMBER(6)	

NORMALIZATION

Additional entities and FDs after ER to schema conversion:
Since the new relations obtained from the ER to schema decomposition (due to multivalued attributes- 1NF) also have to be considered for normalization, we have to derive new FDs for them.

Functional Dependencies for the additional entities:

Order_list:

order_id, product_id->qty_ordered

Supplier_Inventory:

supplier_id, product_id->qty_supplied

Pharmacy_Inventory:

pharmacy_id, product_id->qty_available

(since these new relations simply contain a composite key and a single non-prime attribute, only one FD exists for each)

Pharmacy:

Entity : Pharmacy

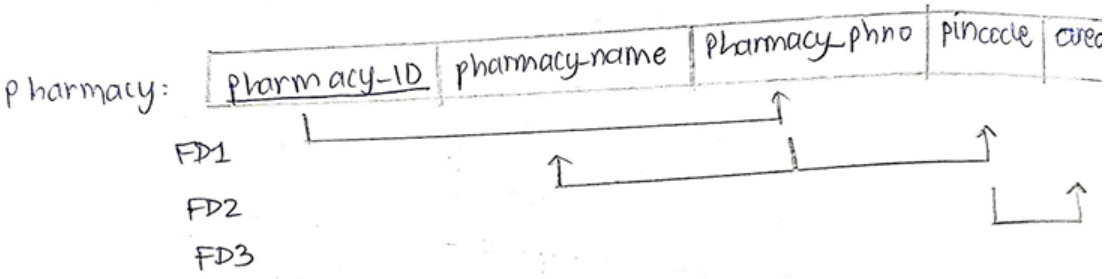
Canonical Cover: Pharmacy_ID ⟶ Pharmacy_phno
Pharmacy_phno ⟶ Pharmacy_name
Pharmacy_phno ⟶ pincode
pincode ⟶ Area

Candidate Key(s) : Pharmacy_ID

Primary Key : Pharmacy_ID

FDs : FD1 : Pharmacy_ID → Pharmacy_Phno
FD2 : Pharmacy_phno → Pharmacy_name, pincode
FD3 : pincode → area

Pharmacy:



1NF : The table is already in 1NF, as there are no multivalued attributes present.

2NF : The table is already in 2NF, as there are no partial functional dependencies present.

FD1 : fully functionally dependent
FD2 : fully functionally dependent
FD3 : fully functionally dependent.

3NF : The table is not in 3NF, as some transitive functional dependencies exist.

(i) Consider FD1 and FD2 :

Pharmacy_ID is a primary key

Pharmacy_phno is not a prime/candidate key (LHS of FD2)

pharmacy_name & pincode are not prime/candidate keys (RHS of FD2)

⇒ 3NF violation : pharmacy_ID → pharmacy_phno

pharmacy_phno → pharmacy_name, pincode.

(By transitivity, this means pharmacy_ID → pharmacy_name, pincode)

∴ Resolving into Pharmacy1 and Pharmacy2 :

Pharmacy1 : | pharmacy_ID | pharmacy_phno |

FD1

Pharmacy2 : | pharmacy_phno | pharmacy_name | pincode | area |

FD2

FD3

(ii) Consider FD2 & FD3 in pharmacy2:

pharmacy_phno is a primary key

pincode is a non-prime key (LHS of FD3)

area is a non-prime key (RHS of FD3)

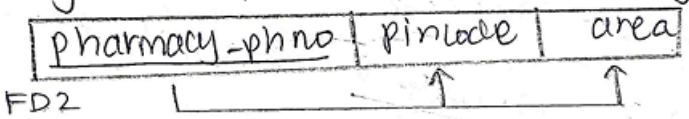
⇒ 3NF violation: pharmacy_phno → pincode

pincode → area

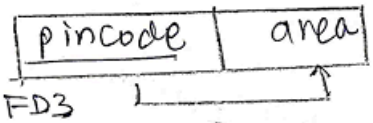
(By transitivity, this means pharmacy_phno → area)

∴ resolving pharmacy 2 into pharmacy 2A & pharmacy2B:

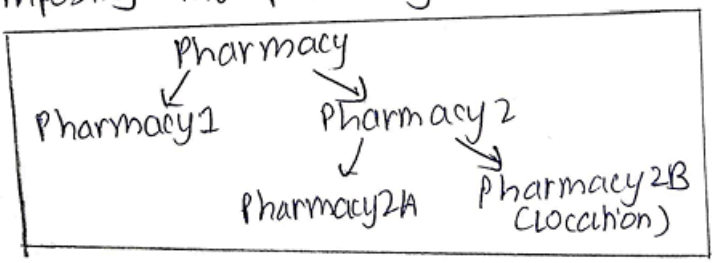
Pharmacy 2A:



Pharmacy 2B: (Location)



Hence, after decomposing the pharmacy table, all FDs are preserved



3NF NORMALIZED.

Customer:

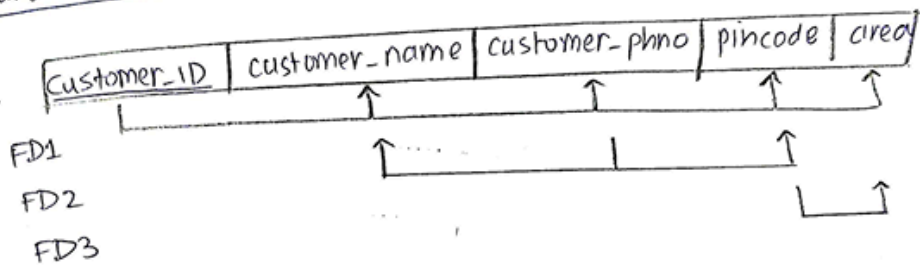
Entity : Customer

Canonical cover : Customer_ID → customer_phno
Customer_phno → customer_name
customer_phno → pincode
pincode → area.

Primary / candidate key : customer_ID

FDs :

Customer:



1NF : ✓ No multivalued attributes

2NF : ✓ FD1 : Fully functionally dependant
FD2 : fully functionally dependant
FD3 : Fully functionally dependant.
No partial functional dependencies exist.

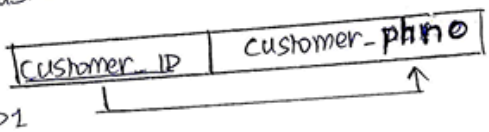
3NF : X transitive FDs exist

i) consider FD1 & FD2 : Customer_ID is PK
customer_phno is not a prime / candidate key.
pincode, customer_name are non-prime keys.

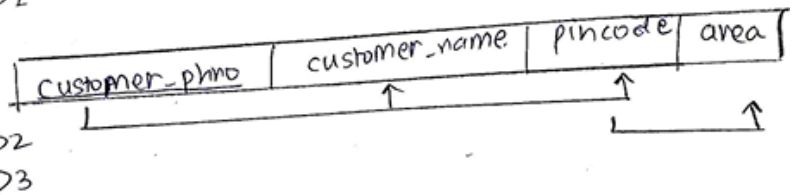
⇒ 3NF violation : customer_ID → customer_phno
customer_phno → customer_name, pincode

∴ Resolving into customer 1 and customer 2

Customer 1:



Customer 2:



(ii) Consider FD2 & FD3 in Customer 2 : Customer_phno is a PK

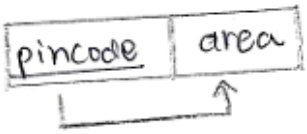
in FD2: pincode is a non-prime attribute
area is a non-prime attribute.

→ 3NF violation: Customer_phno → pincode
pincode → area

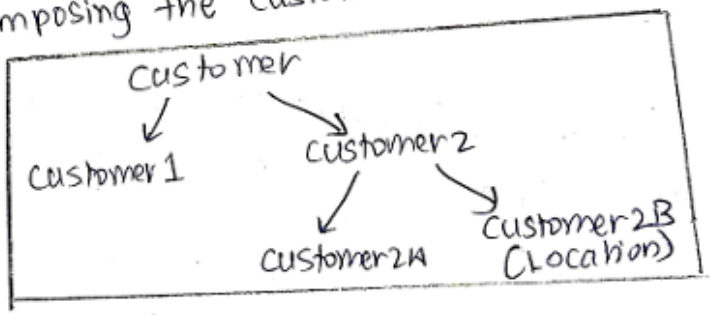
∴ resolving Customer 2 into Customer 2A and Customer 2B:

Customer 2A:

Customer 2B : (Location)



Hence, after decomposing the Customer table, all FDs are preserved



3NF NORMALIZED.

Supplier:

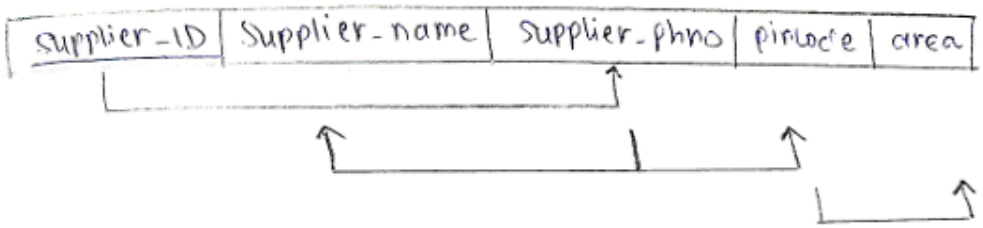
Entity: Supplier

Canonical cover: Supplier_ID → Supplier_phno
Supplier_phno → Supplier_name
Supplier_phno → pincode
pincode → area.

Primary / candidate key(s): Supplier_ID

FDs:

Supplier:



1NF: ✓ No multivalued attributes

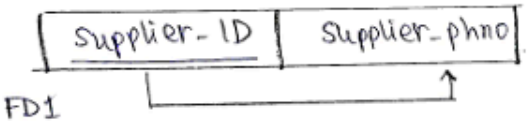
2NF: ✓ No composite prime keys. FD1, FD2, FD3 are fully functionally dependent, no partial FDs

3NF: ✗ transitive FDs exist.

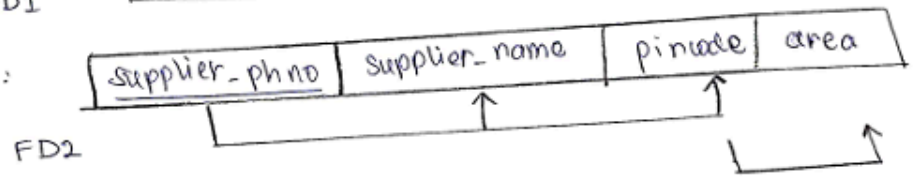
(i) consider FD1 & FD2 : Supplier_ID is PK
Supplier_phno is not a candidate key, and pincode, ~~area~~ Supplier_name are non-prime keys
⇒ 3NF violation: Supplier_ID → Supplier_phno
Supplier_phno → Supplier_name, pincode

∴ Resolving into 2 tables supplier1 & supplier2

Supplier1:



Supplier2:



(ii) Consider FD2 & FD3 (in supplier 2): supplier_phno is a primary key in supplier 1

in FD3: pincode is a non-prime attribute

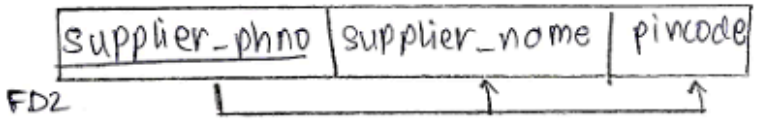
area is a non-prime attribute

⇒3NF violation: supplier_phno → pincode

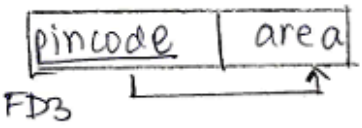
pincode → area

∴ resolving supplier 2 into supplier2A & Supplier 2B:

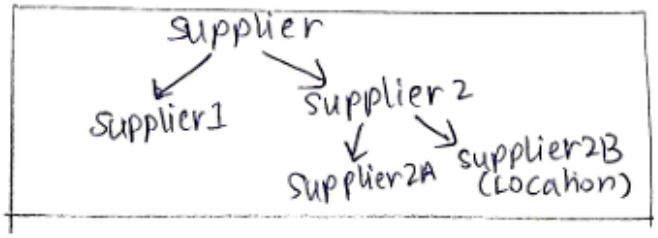
supplier2A:



supplier 2B: (Location)



Hence, after decomposing the Supplier table, all FDS are preserved



3NF NORMALIZED.

Transaction:

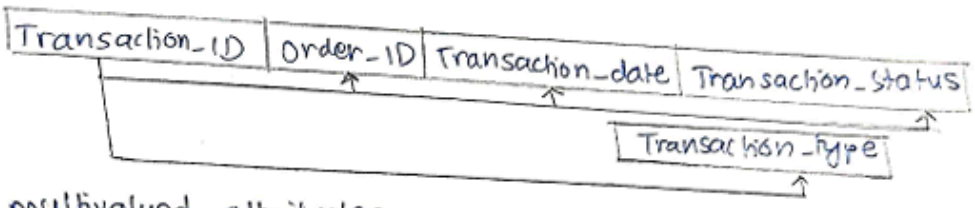
Entity : Transaction

Canonical Cover : Transaction_ID → Transaction_date
Transaction_ID → Transaction_status
Transaction_ID → Transaction_type
Transaction_ID → Order_ID

Primary/Candidate Key : Transaction_ID

FDs : FD1 : Transaction_ID → Transaction_date, Transaction_status, Transaction_type, Order_ID

Transaction :



1NF : ✓ No multivalued attributes

2NF : ✓ No partial functional dependencies.
FD1 : fully functionally dependent.

3NF : ✓ No transitive dependencies

Transaction_ID is a primary key (LHS of FD1)

Transaction_date, Transaction_type, Transaction_status, Order_ID are (RHS of FD1). non-prime attributes.

There is no transitivity.

∴ The Transaction entity is normalized until 3NF.

Order:

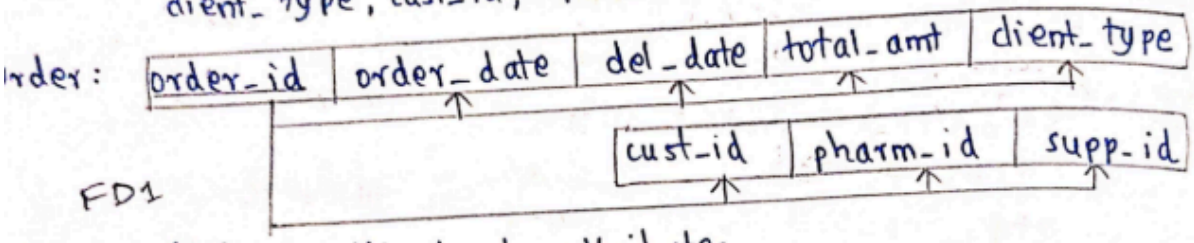
Entity: Order

Canonical cover: order_id → order_date
order_id → del_date
order_id → total_amt
order_id → client_type
order_id → cust_id
order_id → pharm_id
order_id → supp_id

(NOTE: the attributes product_ID, qty-ordered have been removed from the orders table and made into a seperate relation order_item during ER to Schema / INF conversion (multivalued). So related FDs are removed)

Primary, Candidate Key: {order_id}

FDs: FD1: order_id → {order_date, delv_date, total_amt, client_type, cust_id, pharm_id, supp_id}



1NF: ✓ No multivalued attributes

2NF: ✓ No partial functional dependencies
FD1: Fully functionally dependent

3NF: ✓ No transitive dependencies

Order_id is a primary key (LHS of FD1)

order_date, del_date, total_amt, client_type, cust_id, pharm_id, supp_id are non-prime attributes (RHS of FD1)

There is no transitivity

∴ The Order entity is normalised with 3 NF.

Product:

Entity: Products

Canonical Cover: Product_ID → Product_Name
Product_ID → dosage
{Product_Name, dosage} → Product_ID
{Product_Name, dosage} → ~~Supplier_ID~~ exp_date
{Product_Name, dosage} → mfg_date
{Product_Name, dosage} → unit_price
Product_Name → product_category

(NOTE: During ER → schema conversion, attributes pharmacy_ID, supplier_ID, qty_available, qty supplied were removed from the products table (multivalued). ~~Hence, rela~~ and mapped into a new relation pharmacy_inventory & supplier_inventory. Hence, related FDs are removed here).

Candidate keys: Product_ID
{Product_Name, dosage}

Primary Key: Product_ID

FDs: FD1: Product_ID → Product_Name, dosage
FD2: {Product_Name, Dosage} → Product_ID, exp_date, mfg_date, unit_price.
FD3: Product_Name → Product_category.

Products:

Product_ID	Product_Name	Dosage	exp_date	mfg_date	up	prod_categ

FD1
FD2
FD3

INF: The table is already in INF, as there are no multivalued attributes present.

2NF : FD1 : fully functionally dependent
FD2 : fully functionally dependent.
FD3 : partially functionally dependent.
In FD3, Product_Name → product category.
LHS product_Name is a part of a candidate key
RHS product_category is a non-prime attribute.

∴ The table is not in 2NF, partial FDs exist.

∴ Decomposing products into Product 1 and Product 2 :

Product 1 :

Product_ID	Product_Name	Dosage	exp-date	mfd	up

FD1

FD2

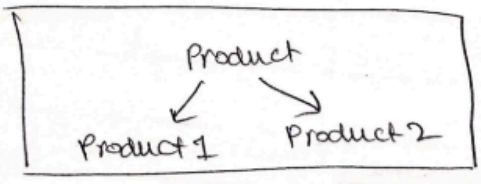
product 2 :

Product_Name	Product_Category

FD3.

Clearly, all FDs are preserved.
∴ 2NF NORMALIZED.

3NF : The table is already in 3NF, as no transitive dependencies exist.
FD1 (in product 1) : no transitive dependency
FD2 (in product 1) : no transitive dependency, as {product_name, dosage} is CK
FD3 (in product 2) : no transitive dependency.



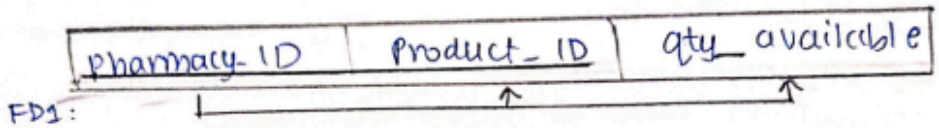
Pharmacy_Inventory:

Entity: Pharmacy_Inventory

Canonical cover: {Pharmacy_ID, Product_ID} → qty_available.

Primary/Candidate key: {Pharmacy_ID, Product_ID}

FDs: FD1: {Pharmacy_ID, Product_ID} → qty_available.



1NF: ✓ no multivalued attributes.

2NF: ✓ No partial dependencies exist.

FD1: fully functionally dependent.

3NF: ✓ No transitive dependencies.

{Pharmacy_ID, Product_ID} is a primary key (LHS of FD1)

qty_available is a non-prime attribute. (RHS of FD1)

There is no transitivity.

∴ Pharmacy_Inventory is normalized until 3NF.

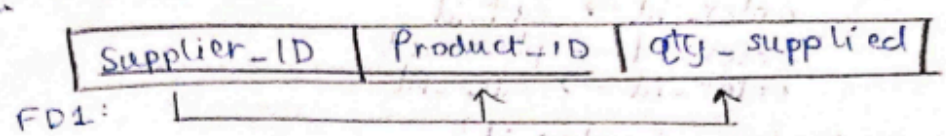
Supplier_Inventory:

Entity, Supplier_Inventory

Canonical cover: {Supplier_ID, Product_ID} → qty_supplied.

Primary/Candidate key: {Supplier_ID, Product_ID}

FDs: FD1: {Supplier_ID, Product_ID} → qty_supplied.



1NF: ✓ No multivalued attributes.

2NF: ✓ No partial dependencies exist:
FD1: fully functionally dependent.

3NF: ✓ No transitive dependencies.
{Supplier_ID, Product_ID} is a primary key (LHS of FD1)
qty_supplied is a non-prime attribute (RHS of FD1).
There is no transitivity.

∴ Supplier_Inventory is normalized until 3NF.

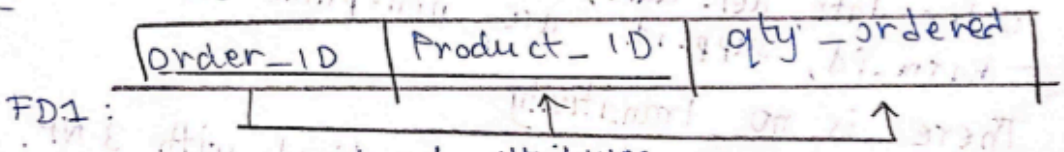
Order_Item:

Entity: Order_Item

Canonical cover: {Order_ID, Product_ID} → qty_ordered.

Primary/candidate key: {Order_ID, Product_ID}

FDs: FD1: {Order_ID, Product_ID} → qty_ordered.



1NF: ✓ no multivalued attributes.

2NF: ✓ No partial dependencies exist.
FD1: fully functionally dependent.

3NF: ✓ No transitive dependencies.

∴ Order_Item is normalized until 3NF.

CONCLUSION

In conclusion, the Pharmacy Supply Chain Management (SCM) system plays a pivotal role in optimizing operations and ensuring efficiency within the pharmaceutical industry. Through a systematic approach encompassing the identification of entities and attributes, establishment of relationships, and development of Entity-Relationship (ER) and schema diagrams, the foundation for effective data management and coordination is established. Additionally, the definition of functional dependencies (FDs) and validation of super keys and primary keys ensure data integrity and relational database efficiency. Moving forward, the implementation and normalization of tables further refines the database structure, enhancing performance and facilitating seamless management of pharmacy supply chains. Overall, the Pharmacy SCM system represents a critical tool for streamlining operations, improving customer service, and driving business success within the pharmaceutical domain.