Phase 1 – E-Commerce Data Model Relationship Mapping

1. Introduction

In Phase 1 of our E-Commerce Marketplace, we created the following collections:

- User
- Admin
- AdminConfig
- Seller
- Customer
- Category
- Product
- Inventory
- Cart
- Wishlist
- Order
- Payment

This document explains:

- How these collections are related
- Which relationships are embedded
- Which relationships are referential
- Why each design decision was taken

2. High-Level Architecture Overview

The system can be divided into four logical layers:

1. Identity Layer
   - User
2. Governance Layer
   - Admin
   - AdminConfig
3. Supply Layer
   - Seller
   - Category
   - Product
   - Inventory
4. Demand & Commerce Layer
   - Customer

- Cart
- Wishlist
- Order
- Payment

3. Identity Layer

3.1 User

User represents authentication and identity.

Relationships:

- One-to-One (Reference) → Customer
- One-to-One (Reference) → Seller
- One-to-One (Reference) → Admin

Design decision:

We used referencing because each of these entities has an independent lifecycle.

Example:

Customer

userId: ObjectId

Seller

userId: ObjectId

Admin

userId: ObjectId

Why reference?

- Separation of concerns
- Clean domain boundaries
- Independent querying

3.2 Relationships Overview

From	To	Type	Strategy	Purpose
User	Customer	1–1	Reference	Buyer profile extension
User	Seller	1–1	Reference	Seller profile extension
User	Admin	1–1	Reference	Admin profile extension
User	AdminConfig	1–N	Reference	Audit (createdBy, updatedBy)
User	Order	1–N	Reference	Audit tracking
User	Payment	1–N	Reference	Audit tracking

4. Governance Layer

4.1 Admin

Admin represents platform operators.

Relationship:

- One-to-One (Reference) → User

Admin is a profile extension of User.

Relationships Overview

From	To	Type	Strategy	Purpose
Admin	User	1–1	Reference	Identity link
Admin	Category	1–N	Reference (audit)	Category management
Admin	Product	1–N	Reference (approval)	Product moderation

4.2 AdminConfig

AdminConfig stores global platform rules.

Relationship:

- Created/Updated by → User (Reference)

AdminConfig is not tied to a specific admin user permanently.

It stores system-level rules like:

- Commission percentage
- Subscription plans

Design decision:

Separate collection to avoid mixing system rules with person data.

Relationships Overview

From	To	Type	Strategy	Purpose
AdminConfig	User	1–N	Reference	Audit tracking
AdminConfig	Custome r	Logical	Service-level	Subscription benefit application

5. Supply Layer

5.1 Seller

Seller represents vendors.

Relationship:

- One-to-One (Reference) → User
- One-to-Many (Reference) → Product
- One-to-Many (Reference) → Inventory

Why reference?

- Seller has independent lifecycle

- Products are large and independent
- Inventory is transaction-heavy

Relationships Overview

From	To	Type	Strategy	Purpose
Seller	User	1–1	Reference	Identity link
Seller	Product	1–N	Reference	Seller owns products
Seller	Inventory	1–N	Reference	Seller stock per location
Seller	OrderItems	1–N	Embedded Snapshot	Revenue tracking

5.2 Category

Category represents product classification.

Relationship:

- Self-reference (Parent Category)

Example:

parentCategory: ObjectId

This supports multi-level category structure.

Design decision:
Referential self-relationship allows unlimited depth.

Relationships Overview

From	To	Type	Strategy	Purpose
Category	Category	1–N	Self Reference	Hierarchical structure
Category	Product	1–N	Reference	Product classification

5.3 Product

Product represents sellable items.

Relationships:

- Many-to-One (Reference) → Seller
- Many-to-One (Reference) → Category
- One-to-Many (Reference) → Inventory
- One-to-Many (Embedded Snapshot in Order)

Why reference Seller and Category?

- Independent entities
- Frequently queried separately

Relationships Overview

From	To	Type	Strategy	Purpose
Product	Seller	N–1	Reference	Ownership
Product	Category	N–1	Reference	Classification
Product	Inventory	1–N	Reference	Stock per location
Product	CartItems	1–N	Reference	Purchase intent
Product	WishlistItems	1–N	Reference	Saved intent
Product	OrderItems	1–N	Embedded Snapshot	Historical record

5.4 Inventory

Inventory represents stock per product per location.

Relationships:

- Many-to-One (Reference) → Product
- Many-to-One (Reference) → Seller

Why reference?

- Inventory changes frequently
- Transaction-heavy
- Independent lifecycle

We did not embed inventory inside Product to avoid large documents and frequent updates.

Relationships Overview

From	To	Type	Strategy	Purpose
Inventory	Product	N–1	Reference	Stock mapping
Inventory	Seller	N–1	Reference	Ownership
Inventory	Order	Logical	Service-level	Stock deduction

6. Demand & Commerce Layer

6.1 Customer

Customer represents buyer profile.

Relationships:

- One-to-One (Reference) → User
- One-to-Many (Embedded) → Addresses
- One-to-Many (Reference) → Order

- One-to-One (Reference) → Cart
- One-to-Many (Reference) → Wishlist

Why embed addresses?

- Addresses belong strictly to Customer
- Always fetched together
- Limited growth

Relationships Overview

From	To	Type	Strategy	Purpose
Customer	User	1–1	Reference	Identity link
Customer	Address	1–N	Embedded	Delivery addresses
Customer	Cart	1–1	Reference	Active cart
Customer	Wishlist	1–N	Reference	Multiple wishlists
Customer	Order	1–N	Reference	Purchase history
Customer	Payment	1–N	Reference	Financial history

6.2 Cart

Cart represents temporary purchase intent.

Relationships:

- One-to-One (Reference) → Customer
- Many-to-One (Reference) → Product
- Many-to-One (Reference) → Seller

Cart items are embedded inside Cart.

Why embed cart items?

- Cart items belong only to that cart
- Always fetched with cart
- No independent lifecycle

Relationships Overview

From	To	Type	Strategy	Purpose
Cart	Customer	1–1	Reference	Ownership
Cart	CartItems	1–N	Embedded	Temporary intent
CartItems	Product	N–1	Reference	Product link
CartItems	Seller	N–1	Reference	Multi-seller support

6.3 Wishlist

Wishlist represents long-term intent.

Relationships:

- Many-to-One (Reference) → Customer
- Many-to-One (Reference) → Product
- Many-to-One (Reference) → Seller

Wishlist items are embedded inside Wishlist.

Why embed?

- Belongs only to that wishlist
- Always fetched together

Relationships Overview

From	To	Type	Strategy	Purpose
Wishlist	Customer	N–1	Reference	Ownership
Wishlist	WishlistItems	1–N	Embedded	Saved products
WishlistItems	Product	N–1	Reference	Product link
WishlistItems	Seller	N–1	Reference	Seller tracking

6.4 Order

Order represents confirmed purchase.

Relationships:

- Many-to-One (Reference) → Customer
- One-to-One (Reference) → Payment
- Many-to-One (Reference) → Product (snapshot)
- Many-to-One (Reference) → Seller (snapshot)

Order Items are embedded.

Why embed order items?

- Order is a historical snapshot
- Items do not exist outside order
- Always fetched together
- Prevents data inconsistency

We also store:

- productName
- priceAtPurchase
- membershipSnapshot

This is intentional denormalization.

Relationships Overview

From	To	Type	Strategy	Purpose
Order	Customer	N–1	Reference	Buyer
Order	OrderItems	1–N	Embedded	Snapshot
OrderItems	Product	N–1	Embedded Snapshot	Historical accuracy
OrderItems	Seller	N–1	Embedded Snapshot	Revenue tracking
Order	Payment	1–1	Reference	Financial link
Order	Inventory	Logical	Service-level	Stock update

6.5 Payment

Payment represents financial transaction.

Relationships:

- Many-to-One (Reference) → Order
- Many-to-One (Reference) → Customer

Refunds are embedded inside Payment.

Why embed refunds?

- Refund is part of payment lifecycle
- Always fetched with payment
- Financial containment

Relationships Overview

From	To	Type	Strategy	Purpose
Payment	Order	N–1	Reference	Financial record
Payment	Customer	N–1	Reference	Payment owner
Payment	Refunds	1–N	Embedded	Refund tracking

7. Summary Table of Relationships

Parent / Source	Related Collection	Cardinality	Strategy	Purpose
User	Customer	1–1	Reference	Customer profile extension
User	Seller	1–1	Reference	Seller profile extension
User	Admin	1–1	Reference	Admin profile extension
User	AdminConfig	1–N	Reference	Audit (createdBy, updatedBy)
Seller	Product	1–N	Reference	Seller owns multiple products
Seller	Inventory	1–N	Reference	Seller stock across locations
Seller	OrderItems	1–N	Embedded Snapshot	Revenue tracking per seller
Category	Category (parentCategory)	1–N	Self Reference	Hierarchical category structure
Category	Product	1–N	Reference	Product classification
Product	Inventory	1–N	Reference	Stock per location
Product	CartItems	1–N	Reference	Purchase intent
Product	WishlistItems	1–N	Reference	Long-term intent
Product	OrderItems	1–N	Embedded Snapshot	Historical record
Customer	Address	1–N	Embedded	Delivery addresses
Customer	Cart	1–1	Reference	Active shopping cart
Customer	Wishlist	1–N	Reference	Multiple wishlists
Customer	Order	1–N	Reference	Purchase history
Customer	Payment	1–N	Reference	Financial history
Cart	CartItems	1–N	Embedded	Temporary purchase intent
Wishlist	WishlistItems	1–N	Embedded	Product bookmarking
Order	OrderItems	1–N	Embedded	Snapshot of purchased products
Order	Payment	1–1	Reference	Financial transaction
Order	Inventory	Logical 1–N	Service-Level	Stock deduction on confirmation
Payment	Refunds	1–N	Embedded	Partial / full refunds

8. Key Design Principles Used in Phase 1

1. Embed when child belongs strictly to parent.
2. Reference when entity has independent lifecycle.
3. Denormalize for historical accuracy.
4. Avoid deep nesting for scalable entities.
5. Keep transactional data separate (Inventory, Payment).

9. Conclusion

In Phase 1, we used a hybrid MongoDB modeling approach:

- Embedding for containment
- Referencing for independence
- Denormalization for performance and audit

This structure is:

- Scalable
- Microservice-ready
- Audit-safe
- Subscription-ready
- Multi-seller ready
- Partial cancellation ready