Optimized MongoDB E-commerce Database Schema

Collections Structure

users

javascript

```
{
  _id: ObjectId, // Primary Key (auto-generated)
  username: String, // unique, indexed
  fullName: String,
  email: String, // unique, indexed
  password: String, // hashed
  role: String, // enum: ["admin", "customer"]
  profile: {
    avatar: String, // URL to image
    phone: String,
    dateOfBirth: Date
  },
  addresses: [{ // Array for multiple addresses
    type: String, // "billing", "shipping", "both"
    street: String,
    city: String,
    state: String,
    postalCode: String,
    country: String,
    isDefault: Boolean
  }],
  isActive: Boolean, // for soft delete
  createdAt: Date,
  updatedAt: Date
}
```

categories

javascript

```
{
  _id: ObjectId, // Primary Key
  name: String, // "Men", "Women", "Unisex", "Electronics", etc.
  slug: String, // URL-friendly version, unique, indexed
  description: String,
  parentId: ObjectId, // Self-reference for subcategories
  level: Number, // 0 for root categories, 1 for subcategories, etc.
  isActive: Boolean,
  createdAt: Date,
  updatedAt: Date
}
```

products

javascript

```
{
  _id: ObjectId, // Primary Key
  name: String, // indexed for search
  slug: String, // URL-friendly, unique, indexed
  description: String,
  shortDescription: String,
  price: Number, // current price
  originalPrice: Number, // for discounts
  categoryId: ObjectId, // Reference to categories
  subcategoryId: ObjectId, // Optional reference to subcategory
  inventory: {
    stock: Number,
    lowStockThreshold: Number,
    isInStock: Boolean // computed field
  },
  images: [String], // Array of image URLs
  specifications: Object, // Flexible object for product specs
  tags: [String], // For search and filtering
  isActive: Boolean,
  isFeatured: Boolean,
  averageRating: Number, // Denormalized for performance
  totalReviews: Number, // Denormalized for performance
  createdAt: Date,
  updatedAt: Date
}
```

orders

javascript

```
{
  _id: ObjectId, // Primary Key
  orderNumber: String, // Human-readable order number, unique, indexed
  customerId: ObjectId, // Reference to users
  items: [{ // Embedded order items for better performance
    productId: ObjectId, // Reference to products
    productName: String, // Snapshot for historical data
    quantity: Number,
    priceAtPurchase: Number,
    totalPrice: Number // quantity * priceAtPurchase
  }],
  pricing: {
    subtotal: Number,
    tax: Number,
    shipping: Number,
    discount: Number,
    total: Number
  },
  shippingAddress: {
    street: String,
    city: String,
    state: String,
    postalCode: String,
    country: String
  },
  billingAddress: {
    street: String,
    city: String,
    state: String,
    postalCode: String,
    country: String
  },
  status: String, // "pending", "confirmed", "processing", "shipped", "delivered", "cancelled"
  paymentDetails: {
    method: String, // "credit_card", "paypal", "stripe", etc.
    transactionId: String,
    status: String, // "pending", "completed", "failed", "refunded"
    paidAt: Date
  },
  tracking: {
    carrier: String,
    trackingNumber: String,
    estimatedDelivery: Date
```

```
  },
  createdAt: Date,
  updatedAt: Date
}
```

cart

javascript

```
{
  _id: ObjectId, // Primary Key
  userId: ObjectId, // Reference to users, unique indexed
  items: [{
    productId: ObjectId, // Reference to products
    quantity: Number,
    addedAt: Date
  }],
  updatedAt: Date,
  expiresAt: Date // TTL index for automatic cleanup
}
```

reviews

javascript

```
{
  _id: ObjectId, // Primary Key
  productId: ObjectId, // Reference to products, indexed
  userId: ObjectId, // Reference to users, indexed
  orderId: ObjectId, // Reference to orders (ensures verified purchase)
  rating: Number, // 1-5
  title: String,
  comment: String,
  images: [String], // Review images
  isVerifiedPurchase: Boolean,
  helpfulVotes: Number, // For review ranking
  status: String, // "pending", "approved", "rejected"
  createdAt: Date,
  updatedAt: Date
}
```

wishlist (Additional Collection)

```javascript
{
  _id: ObjectId,
  userId: ObjectId, // Reference to users, unique indexed
  products: [ObjectId], // Array of product IDs
  updatedAt: Date
}
```

Key Optimizations Made

1. Embedded vs Referenced Data

- **Orders**: Embedded order items instead of separate collection for better read performance
- **Users**: Embedded addresses array to handle multiple addresses
- **Categories**: Added hierarchical structure with parentId for subcategories

2. Indexing Strategy

```javascript
```

```
// User indexes
db.users.createIndex({ email: 1 }, { unique: true })
db.users.createIndex({ username: 1 }, { unique: true })
db.users.createIndex({ role: 1 })

// Product indexes
db.products.createIndex({ categoryId: 1 })
db.products.createIndex({ name: "text", description: "text", tags: "text" }) // Text search
db.products.createIndex({ slug: 1 }, { unique: true })
db.products.createIndex({ isActive: 1, isFeatured: 1 })
db.products.createIndex({ price: 1 })
db.products.createIndex({ averageRating: -1 })

// Order indexes
db.orders.createIndex({ customerId: 1, createdAt: -1 })
db.orders.createIndex({ orderNumber: 1 }, { unique: true })
db.orders.createIndex({ status: 1 })
db.orders.createIndex({ "paymentDetails.status": 1 })

// Cart indexes
db.cart.createIndex({ userId: 1 }, { unique: true })
db.cart.createIndex({ expiresAt: 1 }, { expireAfterSeconds: 0 }) // TTL

// Review indexes
db.reviews.createIndex({ productId: 1, createdAt: -1 })
db.reviews.createIndex({ userId: 1 })
db.reviews.createIndex({ productId: 1, userId: 1 }, { unique: true }) // One review per user per product

// Category indexes
db.categories.createIndex({ slug: 1 }, { unique: true })
db.categories.createIndex({ parentId: 1 })
```

3. **Data Consistency & Performance**

- Added denormalized fields (`averageRating`, `totalReviews`) for better read performance
- Included snapshot data in orders for historical accuracy
- Added `isActive` flags for soft deletes
- Implemented proper data types (ObjectId for references)

4. **Additional Features**

- Order numbering system for customer-friendly references

- Hierarchical categories for better organization
- Wishlist functionality
- Review verification through order references
- TTL index on cart for automatic cleanup
- Multiple address support for users

5. **Schema Relationships**

```
users (1) ←→ (many) orders
users (1) ←→ (1) cart
users (1) ←→ (many) reviews
users (1) ←→ (1) wishlist

products (many) ←→ (1) categories
products (1) ←→ (many) reviews
products (many) ←→ (many) orders.items

categories (1) ←→ (many) categories (self-reference)
```

This optimized schema provides better performance, scalability, and maintains data integrity while supporting common e-commerce operations efficiently.