ShopSmart Detailhandelssystem (E-Handel) Projekt

Indholdsfortegnelse:

1.Select a business or organization

1.1 Valg af *business* eller *organization*

For dette projekt er der valgt en fiktiv e-handelsvirksomhed ved navn ShopSmart, som fungerer som en online platform for køb og salg af elektronik og relaterede produkter. Virksomheden opererer med store mængder af forskelligartede data, herunder:

- Produktkataloger
- Produktanmeldelser
- Brugerinformationer
- Ordrehistorik
- Indkøbskurvdata

E-handel er valgt, fordi det repræsenterer en branche med naturligt høj datakompleksitet og krav til hurtig adgang til forskellige typer af data. Databaseløsninger er afgørende for virksomhedens drift. Både for kundevendte funktioner (som produktsøgning og anmeldelser) og interne processer (som ordrebehandling og lagerstyring).

Formålet med projektet har været at undersøge, designe og implementere en polyglot persistence-arkitektur, hvor forskellige datatyper håndteres af forskellige specialiserede databaser:

- **SQL Server** til strukturerede og relationelle data (produkter, ordrer, brugere)
- **MongoDB** til ustrukturerede og fleksible data (anmeldelser og produktspecifikationer)
- **Redis** til hurtige nøglebaserede operationer (midlertidige indkøbskurve)

Denne opsætning simulerer en realistisk løsning anvendt i moderne systemer, hvor man udnytter styrkerne ved forskellige databaseteknologier i en samlet arkitektur.

1.2 Datasæt

Kilde:
Datasættet stammer fra Kaggle og kan findes her:
https://www.kaggle.com/datasets/karkavelrajaj/amazon-sales-dataset

Dette datasæt indeholder detaljerede oplysninger om **1.000+ produkter fra Amazon**, herunder priser, kategorier, ratings og brugeranmeldelser. Det repræsenterer en realistisk sammensætning af både struktureret og ustruktureret data, hvilket gør det ideelt til et **polyglot persistence-projekt**.

1.2.1 Datastruktur og Felter

Felt	Beskrivelse
product_id	Unik ID for hvert produkt
product_name	Navnet på produktet
category	Kategori som produktet tilhører
discounted_price	Nedsat pris (vises for kunden)
actual_price	Den oprindelige pris før rabat
discount_percentage	Rabatsats i procent
rating	Gennemsnitlig rating fra brugere (fx 4.3)
rating_count	Antal brugere, der har afgivet en rating
about_product	Produktbeskrivelse (tekstfelt)
user_id	ID på brugeren, der skrev en anmeldelse
user_name	Navn på brugeren
review_id	Unik ID for anmeldelsen
review_title	Titel/kort beskrivelse af anmeldelsen
review_content	Den fulde anmeldelse (lang tekst)
img_link	Link til produktets billede
product_link	Link til produktet på Amazons hjemmeside

1.2.2 Formål og Relevans i ShopSmart

Vi har udvalgt dette datasæt, da det giver os mulighed for at:

- Importere **produktdata** (navn, pris, kategori, brand) til SQL Server
- Importere **produktanmeldelser og specifikationer** til MongoDB (reviews/specs)
- Øve integrationen mellem forskellige datatyper og databaser
- Skabe en **realistisk brugeroplevelse**, hvor man kan browse produkter, læse anmeldelser og gennemføre ordrer

1.2.3 Eksempler på brug i vores projekt

ShopSmart Funktion	Anvendte Felter Fra Datasættet
Produktkatalog (SQL)	product_id, product_name, actual_price, category
Anmeldelser (MongoDB)	review_id, user_name, review_title, review_content,

	rating
Produktdetaljer (MongoDB)	about_product, img_link, discount_percentage

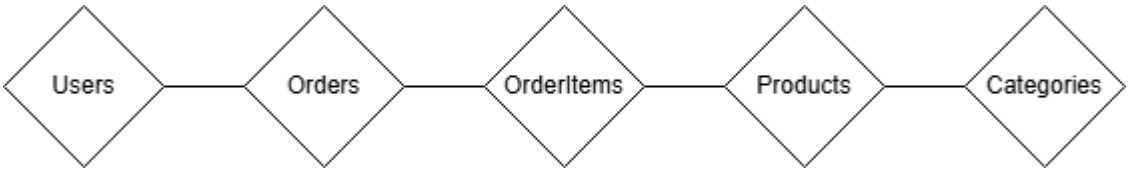
1.3 Teknologier

	Version
VS Code	1.100.2
SSMS	20.1
Docker	20.10.21
MongoDB Compass	1.46.3
Python	3.10
Redis	Redis 7

2. Overblik over entiteter

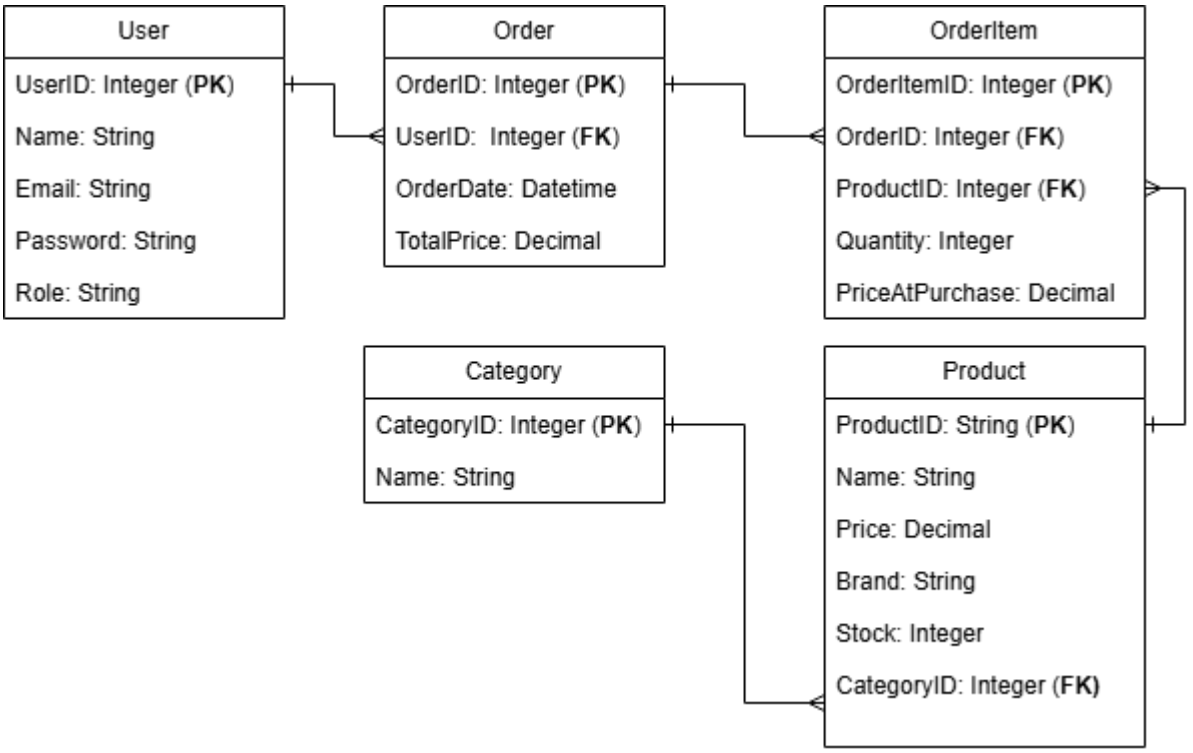
- Products, Users, Orders OrderItems, Reviews, Categories,
    - User: UserID, Name, Email, Password, Role
    - Products: ProductID, Name, Price, Stock (Amount of products), CategoryID
    - Categories: CategoryID, Name
    - Orders: OrderID, UserID, OrderDate, TotalPrice
    - OrderItems: OrderDetailID, OrderID, ProductID, Quantity
    - Reviews: ProductID, UserID, Rating, Comment, Timestamp

2.1 Database design



2.1.1 ER-diagrambeskrivelse - ShopSmart relational model

Diagrammet ovenfor viser den relationelle datamodel, som anvendes i SQL Server-komponenten i ShopSmart-applikation. Det indeholder de centrale entiteter og deres relationer, designet til at understøtte strukturerede forretningsprocesser som ordrebehandling, projektstyring og brugerregistrering.



2.2 Udvidet ERD Diagramforklaring

Diagrammet ovenfor viser den relationelle datamodel, der er designet i SQL Server til ShopSmart-applikationen. Den understøtter kernefunktioner i en e-handelsplatform, såsom brugerhåndtering, produktkatalog, ordrehistorik og lagerstyring.

2.2.1 Tabel-for-tabel forklaring

User (Bruger)	
	Indeholder information om alle systemets brugere, både kunder og administratorer
Felter	• **UserID** (Primær nøgle): Entydig identifikation. • **Name**, **Email**, **Password**: Brugerens loginoplysninger. • **Role**: Adgangsniveau (fx "user" eller "admin").
Relationer	Én bruger kan have mange ordrer (**1:N** til **Order**).

Order (Ordre)	
Beskrivelse	Repræsenterer en enkelt ordre placeret af en bruger.
Felter	• **OrderID** (Primær nøgle): Automatisk genereret ordrenummer. • **UserID** (Fremmednøgle): Refererer til brugeren. • **OrderDate**: Dato og tidspunkt for bestilling. • **TotalPrice**: Samlet beløb for ordren.
Relationer	Hver ordre kan indeholde flere ordrelinjer (1:N til OrderItem).

OrderItem (Ordrelinje)	
Beskrivelse	Repræsenterer en varelinje i en ordre.
Felter	• **OrderItemID** (Primær nøgle): Automatisk genereret linje-ID. • **OrderID** (Fremmednøgle): Refererer til ordren. • **ProductID** (Fremmednøgle): Refererer til det bestilte produkt. • **Quantity**: Antal enheder bestilt. • **PriceAtPurchase**: Pris på købstidspunktet.
Relationer	• Giver mulighed for præcis historik, selv hvis priser ændrer sig

	over tid. • Bruges til kvitteringer og lagertræk.

Product (Produkt)	
Beskrivelse	Indeholder information om de produkter, der sælges i webshoppen.
Felter	• **ProductID** (Primær nøgle): Produktkode (fx SKU). • **ProductName**, **Brand**, **Price**, **Stock**: Beskrivelse og lagerinformation. • **CategoryID** (Fremmednøgle): Tilknytning til en kategori.
Relationer	Hvert produkt tilhører én kategori (N:1 til Category).

Category (Kategori)	
Beskrivelse	Indeholder oversigt over produkttyper, fx “Elektronik”, “Bøger”.
Felter	• CategoryID (Primær nøgle) • CategoryName: Navn på kategorien.
Relationer	Gruppering og filtrering af produkter.

2.3 Designovervejelser

- **Normalisering**: Skemaet er normaliseret til 3NF (Tredje Normalform), hvilket reducerer redundans.
- **Fremmednøgler**: Sikrer dataintegritet og muliggør korrekte JOINs i SQL.

2.4 Relationer

1) User ↔ Order (One-to-Many relation)
   - En bruger (User) kan afgive flere ordrer, men hver ordre tilhører én specifik bruger (User).

2) Order ↔ OrderItem (One-to-Many relation)
   - Hver ordre kan indeholde flere genstande/varer, og hver vare registreres som en række i OrderItem-tabellen.

3) OrderItem ↔ Product (One-to-Many relation)
   - Hvert produkt (Product) kan optræde i flere ordrelinjer (OrderItem), især når det bliver købt af forskellige brugere (User).

4) Product ↔ Category (Many-to-One relation)

- Hvert produkt (Product) tilhører én kategori (Category), men en kategori kan indeholde mange produkter.

Relevans i Polyglot Arkitektur

Denne relationelle model udgør fundamentet i ShopSmart-systemet:

- Den sikrer datakonsistens for centrale forretningsdata som brugere, ordrer og lager.
- Den kobles til **MongoDB** via **ProductID** for at hente fleksible specifikationer og anmeldelser.
- Den fungerer sammen med **Redis** til **sessioner** og **kurvdata**, som senere **gemmes permanent i SQL**.

3. MongoDB

3.1 Beskrivelse af MongoDB

MongoDB sørger for at håndtere brugeranmeldelser (User reviews).

3.2 Formål med MongoDB i systemet

MongoDB bruges til at håndtere brugeranmeldelser, fordi denne type data er:

- Ustruktureret og varierer i længde (kommentarer, stjerner osv.)
- Hyppigt tilføjet/ændret af brugere (reviews kommer løbende)
- Velegnet til søgning, filtrering og gennemsnitsberegning (f.eks. rating)

Vi har valgt at bruge refereret dokumentstruktur, fordi anmeldelser ikke bør gemmes direkte inde i produkter:

- Et produkt kan have mange anmeldelser.
- Anmeldelser skal sorteres, aggregeres og hentes separat.

3.3 MongoDB indhold

Entiteter

Reviews: productID, username, reviewTitle, reviewText, rating, reviewDate

3.4 Review – Dokument Design

	Review (Dokument Design)

Eksempel	{
	“productID”: B07JW9H4J1,
	“username”: “Manav”,
	“reviewTitle”: “Satisfied,Charging is really fast,Value for money,Product review,Good …”,
	“reviewText”: “Looks durable Charging is fine tooNo complains,Charging is really fast…",
	“rating”: 4.2,
	“reviewDate”: "2022-11-10"
	}

Forklaring	**productId**: linkes til SQL data

Bilag

1) T-SQL Tabel Opsætning Kode:

	T-SQL

```
-- Use the ShopSmart database
USE shopsmart;
GO

-- Drop tables if they already exist (for reset)
IF OBJECT_ID('OrderItem', 'U') IS NOT NULL DROP TABLE
OrderItem;
IF OBJECT_ID('[Order]', 'U') IS NOT NULL DROP TABLE
[Order];
IF OBJECT_ID('Product', 'U') IS NOT NULL DROP TABLE
Product;
IF OBJECT_ID('Category', 'U') IS NOT NULL DROP TABLE
Category;
IF OBJECT_ID('[User]', 'U') IS NOT NULL DROP TABLE
[User];
GO

-- Create User table
CREATE TABLE [User] (
    UserID INT PRIMARY KEY,
    Name NVARCHAR(100) NOT NULL,
    Email NVARCHAR(100) NOT NULL UNIQUE,
     Password NVARCHAR(256) NOT NULL,
    Role NVARCHAR(50) NOT NULL CHECK (Role IN ('user',
'admin'))
);
GO

-- Create Category table
CREATE TABLE Category (
    CategoryID INT PRIMARY KEY,
    CategoryName NVARCHAR(100) NOT NULL
);
GO

-- Create Product table
CREATE TABLE Product (
    ProductID NVARCHAR(50) PRIMARY KEY,
    ProductName NVARCHAR(500) NOT NULL,
    Price DECIMAL(10, 2),
     Brand NVARCHAR(255),
     Stock INT NOT NULL,
```

```
    CategoryID INT NOT NULL,
    FOREIGN KEY (CategoryID) REFERENCES
Category(CategoryID)
);
GO

-- Create Order table
CREATE TABLE [Order] (
    OrderID INT PRIMARY KEY IDENTITY(1,1),
    UserID INT NOT NULL,
    OrderDate DATETIME NOT NULL DEFAULT GETDATE(),
    Total DECIMAL(10, 2) NOT NULL CHECK (Total >= 0),
    FOREIGN KEY (UserID) REFERENCES [User](UserID)
);
GO

-- Create OrderItem table
CREATE TABLE OrderItem (
    OrderItemID INT PRIMARY KEY IDENTITY(1,1),
    OrderID INT NOT NULL,
    ProductID NVARCHAR(50) NOT NULL,
    Quantity INT NOT NULL CHECK (Quantity > 0),
    PriceAtPurchase DECIMAL(10, 2) NOT NULL CHECK
(PriceAtPurchase >= 0),
    FOREIGN KEY (OrderID) REFERENCES [Order](OrderID),
    FOREIGN KEY (ProductID) REFERENCES
Product(ProductID)
);
GO
```