Project in Database Design I (1DL301)

Fall 2023, Period 1

Group 39

Nataliia Hordovska
gordovska12@gmail.com

Sareh Jalalizad
sareh.jalalizad.2454@student.uu.se

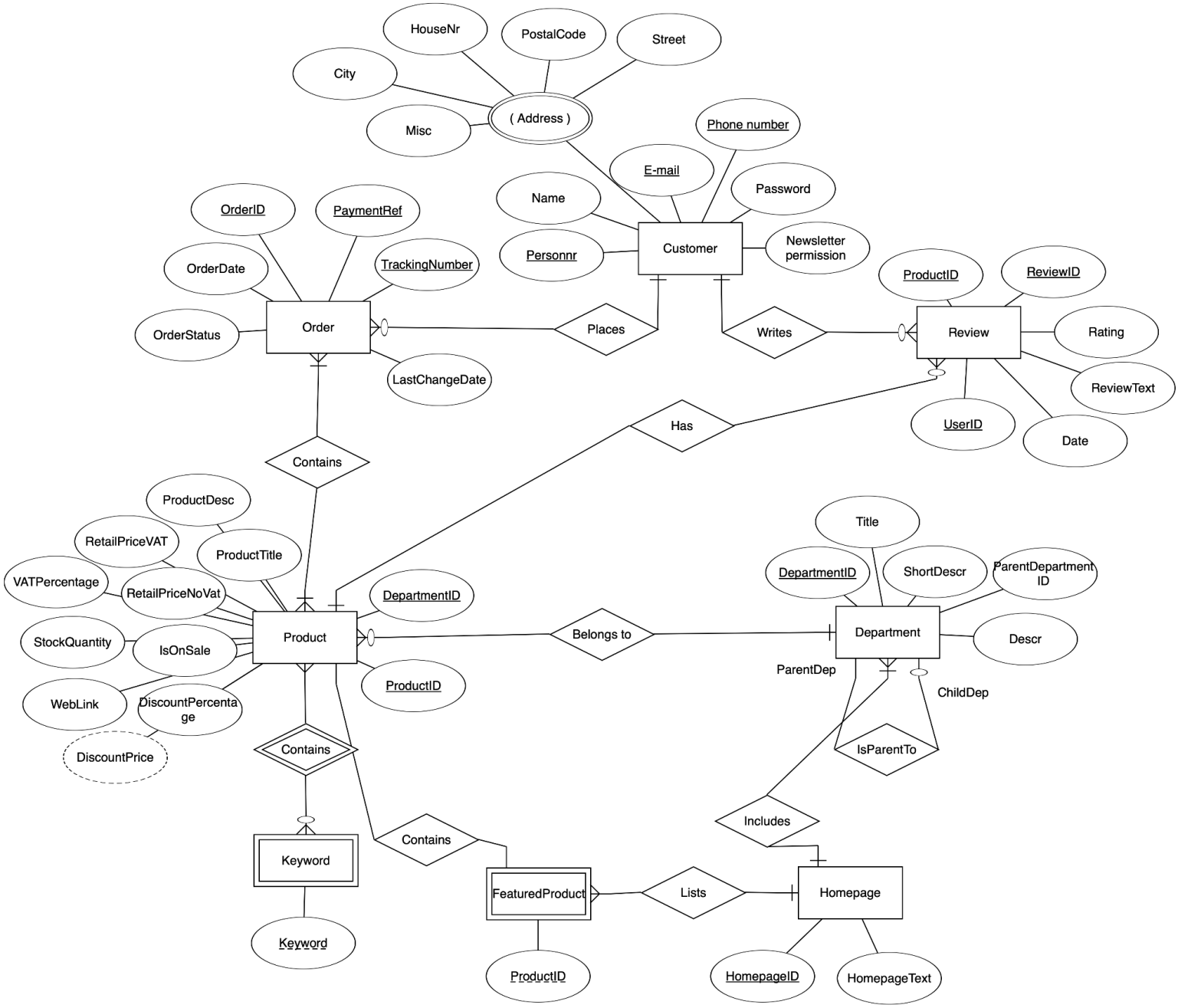
Zakarie Warsame
zakariew11@hotmail.com

Contents:

Part I. Conceptual modeling.

Aim: to develop an ER model from given specifications.

Preliminary result is shown below.



Discussions.

1. **Defining entities.**
   From the project description, we decided to store these entities in the database:
   - Customer;
   - Order;
   - Review;
   - Product;
   - Department;
   - Homepage;
   - FeaturedProduct is a weak entity since it's related to an existing product rather than being a unique one;
   - Keyword as a weak entity identified by the entity 'Product' it is associated with.

2. **Defining relationships.**

1. Department can have zero or one ParentDepartment (ParentDepartmentID in Department).
2. Department can have zero or more ChildDepartments (ParentDepartmentID in Department).
3. Product can be associated with zero or more Keywords ('contains').
4. Customer can place zero or more Orders (CustomerID in Order).
5. Order can contain one or more Products. Here we apply mandatory participation constraint since Order is not created until it contains at least one Product.
6. Customer can write zero or more Reviews (CustomerID in Review).
7. Product can have zero or more Reviews (ProductID in Review).

8. Homepage is associated with one Department (HomepageID in Department).

3. Further discussions.

1) Customer.

We define ‘PersonNr’ as the primary key, since it’s unique and doesn’t change with time. Another possible key could be email and/or phone number, but these values are dynamic, so quite unreliable.

‘Address’ is a composite multivalued attribute. The sub-attributes to ‘Address’ are all the components of a full address, including ‘Misc’, which is a comment section for delivery instructions (door code, leave at the entrance or carport etc.). We decided to make it a multi-value attribute so AltOnline has the same feature as Amazon: choosing in between stored addresses. For instance Customer wants something to be delivered to their workplace instead of home, so it could be executed and the workplace address will be stored along with the delivery instructions.

A Customer can either place an Order or write a Review, so we’re moving on to these entities.

2) Order

OrderID is set as the primary key here. Candidate keys are payment reference number and tracking number. But in order to save space we might use OrderID as PK. An Order contains Product.

3) Product

PK is ProductID, it might be the article number listed by the manufacturer, they are unique for each product. We decided that we don't have to store discounted prices directly if the product is on sale. Hence the derived attribute 'DiscountPrice', which should be derived from RetailPriceVAT and DiscountPercentage. And to keep the DB structured, we insert a foreign key 'DepartmentID'. It's also possible to list RetailPriceVAT as derived from VATPercentage and RetailPriceNoVAT. The product can have zero or more Reviews and contains one or more Keywords. It also belongs to one leaf Department.

4) Keyword

We defined it as a weak entity since it depends exclusively on the product. The relationship is n:m, since one Product may have lots of Keywords, and Products also share Keywords.

5) Review.

PK: ReviewID

FKs: UsedID and ProductID.

6) Department.

PK: DepartmentID.

To represent the hierarchical structure of the Departments there is a recursive relationship IsParentTo and an attribute ParentDepartmentID. So only top-level departments will have a 'null' value in ParentDepartmentID. Top-level Departments are included on the Homepage.

7) Homepage

PK: HomepageID.

Homepage also lists Featured Products.

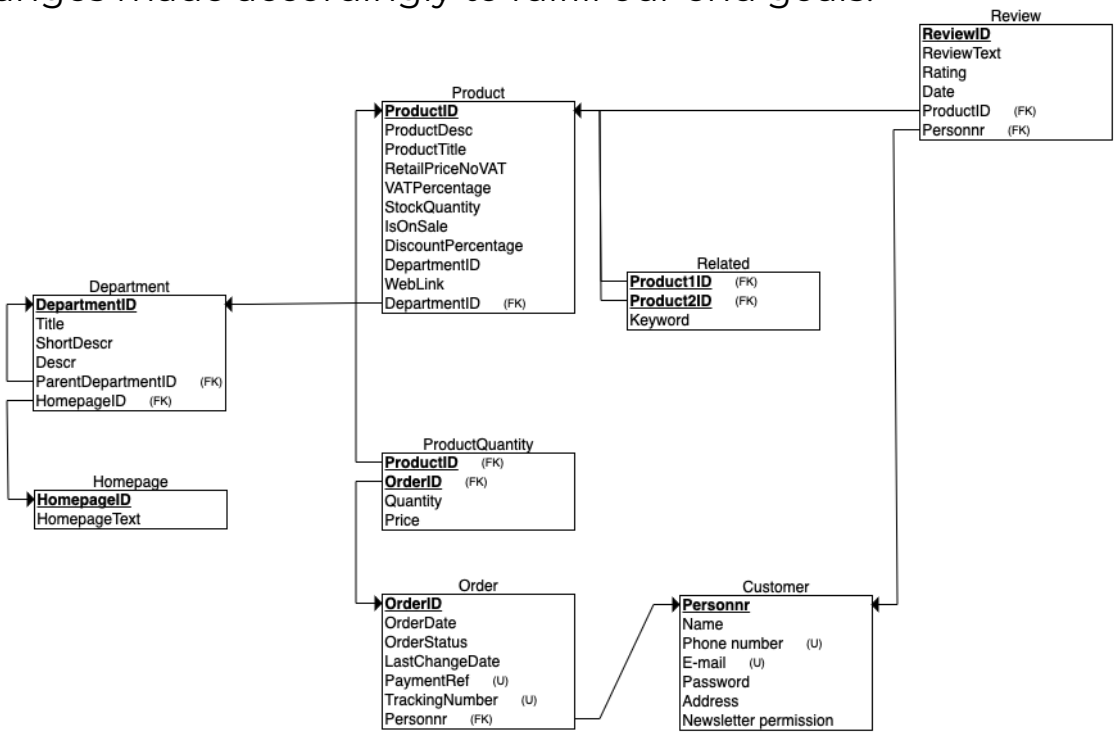
8) FeaturedProduct.

It’s an entity (probably a weak one) with only a FK ProductID from the Product entity.

Part 2. Normalizing ER model up to 3NF

Aim: to normalize the ER model to 3NF from the given specifications.

Below is the relational schema generated by ERDplus, with changes made accordingly to fulfill our end goals:



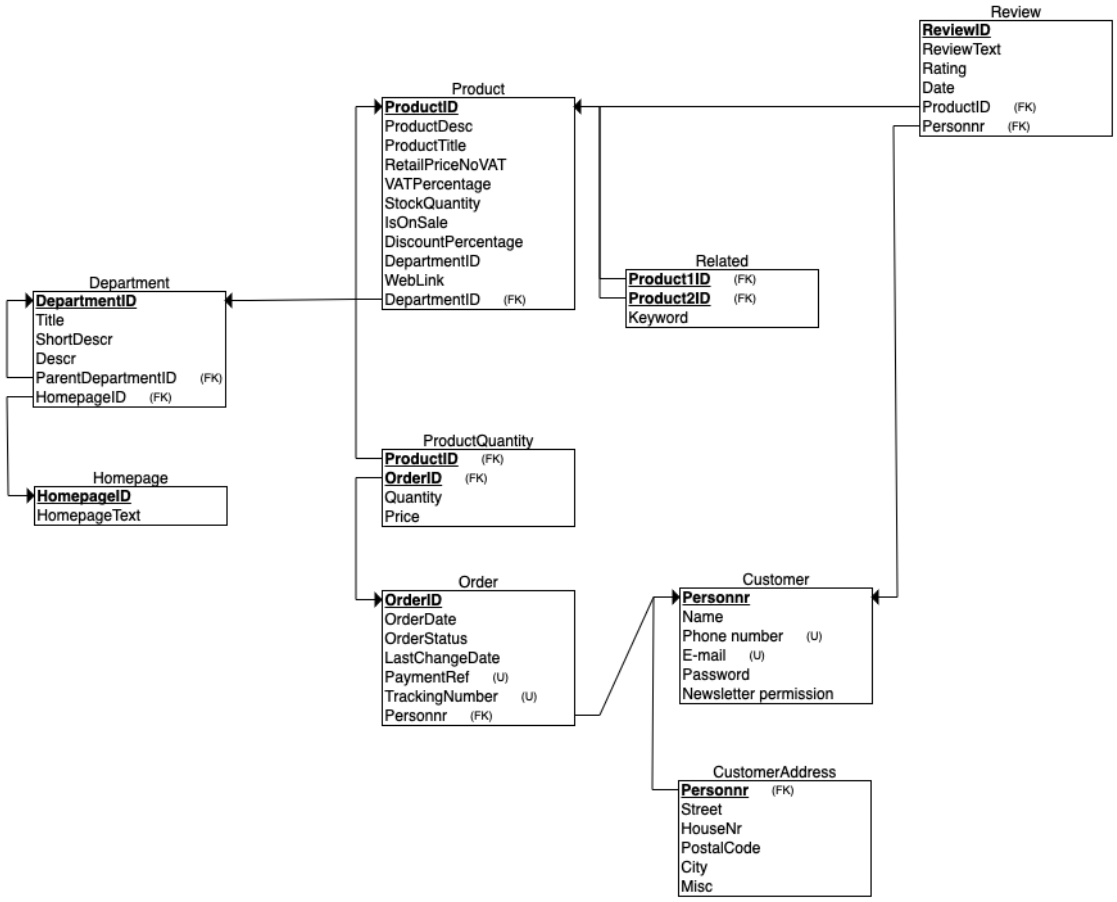
The changes made consist of:

- Removing redundant attributes, such as RetailPriceVAT and isOnSale since these can be derived from RetailPriceNoVAT and if DiscountPercantage is nonzero.
- ProductQuantity was added to represent how many of each product is in each order. Here ProductID and OrderID are both primary keys, since we only care about one specific product in a specific order.
- FeaturedProduct was removed since it doesn't differ from a regular product and therefore unnecessary to store.

- UserID in ‘Review’ was replaced by the foreign key ‘Personnr’.

1. 1NF

To normalize to 1NF we need to check that all attributes in the diagram are atomic. Since ‘Address’ does not contain only atomic value, we need to resolve that. Therefore we need to split it out of ‘Customer’ and create a new table called ‘CustomerAddress’. Now this is our relational schema in 1NF:



2. 2NF

To normalize to 2NF, we need to make sure that every non-prime attribute in each relation is fully functionally dependent on the primary key.

Customer

- Primary Key: Personnr
- Foreign Key: OrderID, ProductID

Here all non-prime attributes fully depend on ‘Personnr’, so it is already in 2NF.

CustomerAddress

- Primary Key: Personnr
- Foreign Key: Personnr

All non-prime attributes fully depend on ‘Personnr’, so it is already in 2NF.

Review

- Primary Key: ReviewID
- Foreign Key: ProductID, Personnr

All non-prime attributes fully depend on ‘ReviewID’, so it is already in 2NF.

Order

- Primary Key: OrderID
- Foreign Key: Personnr

All non-prime attributes fully depend on ‘OrderID’.

ProductQuantity

- Primary Key: OrderID, ProductID
- Foreign Key: OrderID, ProductID

{OrderID, ProductID} -> Quantity is a full FD since neither OrderID -> Quantity nor ProductID-> Quantity hold.

Product

- Primary Key: ProductID
- Foreign Key: DepartmentID

.All non-prime attributes fully depend on ‘OrderID’.

Department

- Primary Key: DepartmentID
- Foreign Key: DepartmentID

All non-prime attributes depend on ‘DepartmentID’

Related

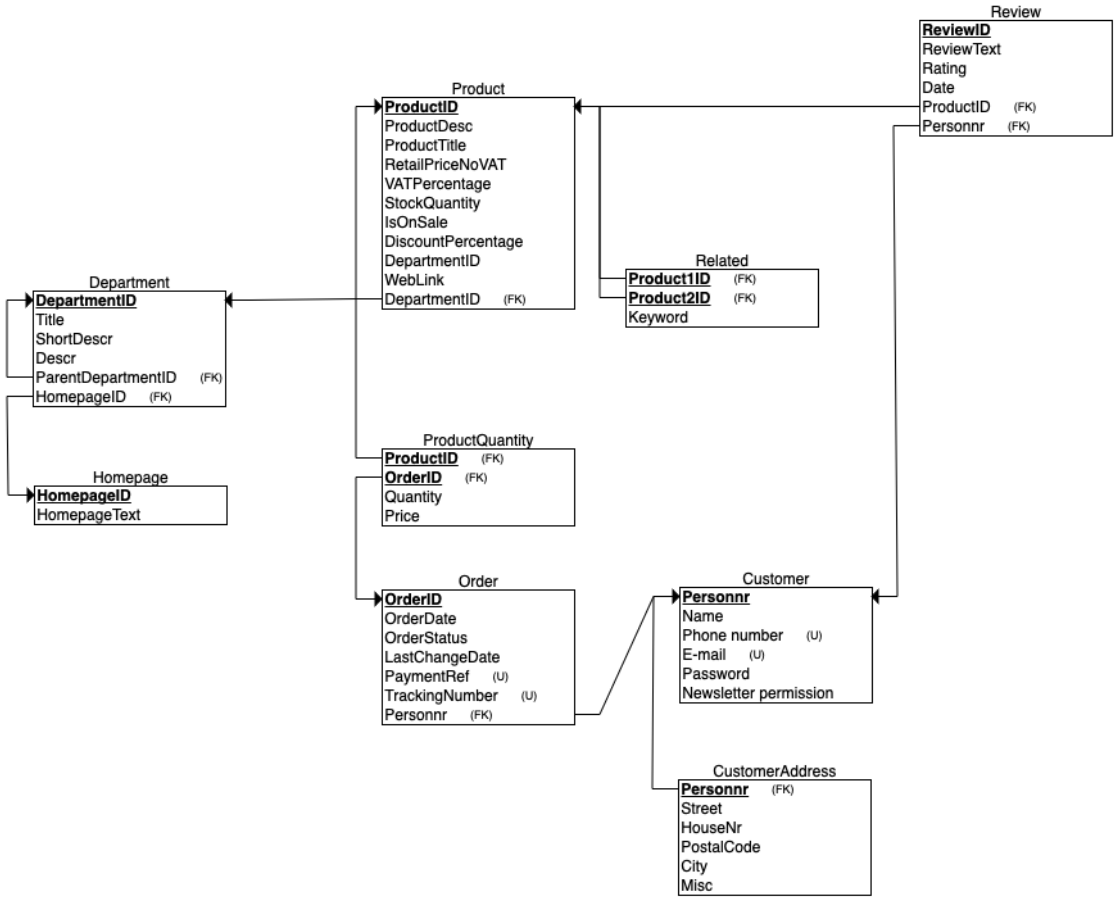
- Primary Key: Product1ID, Product2ID
- Foreign Key: Product1ID, Product2ID

The only non-prime attribute ‘Keyword’ fully depends on the primary keys.

Homepage

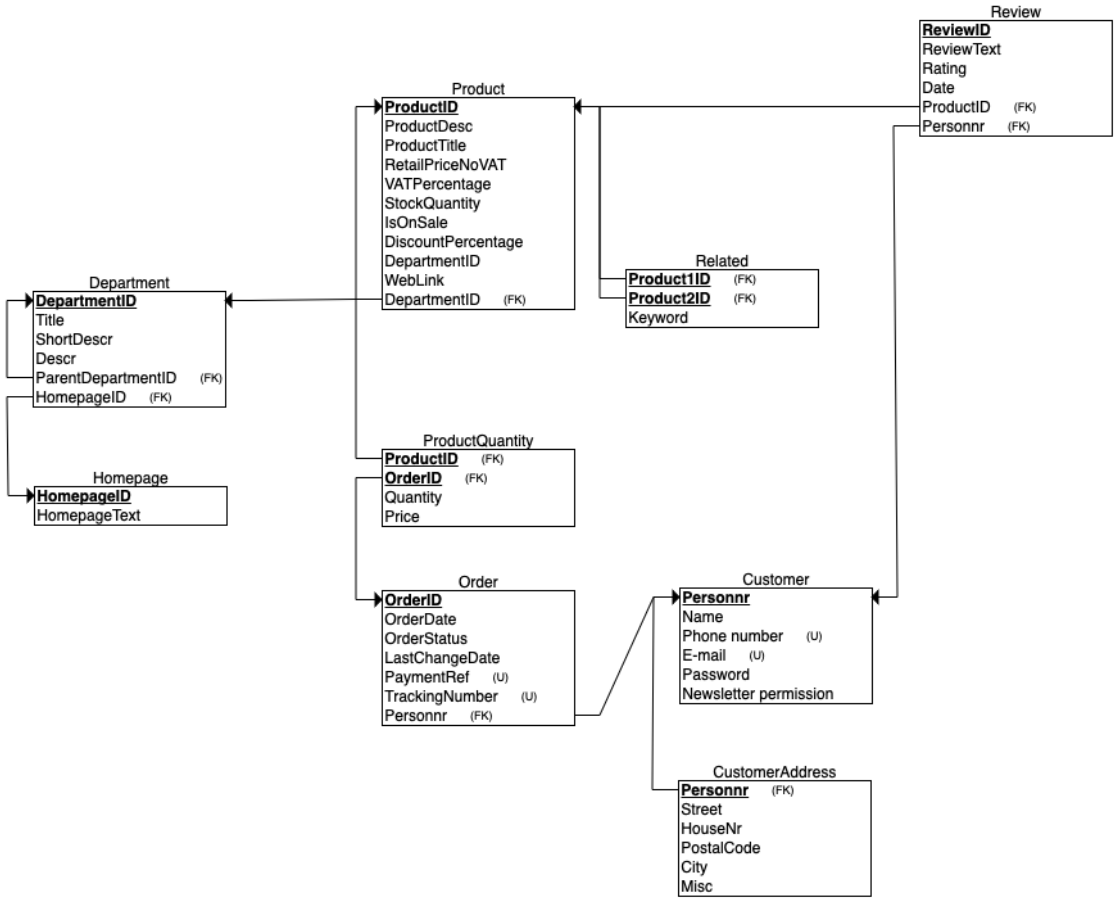
- Primary Key: HomepageID
- Foreign Key: None

All non-prime attributes depend on ‘HomepageID’


Review
ReviewID
ReviewText
Rating
Date
ProductID (FK)
Personnr (FK)
Product
ProductID
ProductDesc
ProductTitle
RetailPriceNoVAT
VATPercentage
StockQuantity
IsOnSale
DiscountPercentage
DepartmentID
WebLink
DepartmentID (FK)
Related
Product1ID (FK)
Product2ID (FK)
Keyword
Department
DepartmentID
Title
ShortDescr
Descr
ParentDepartmentID (FK)
HomepageID (FK)
ProductQuantity
ProductID (FK)
OrderID (FK)
Quantity
Price
Homepage
HomepageID
HomepageText
Order
OrderID
OrderDate
OrderStatus
LastChangeDate
PaymentRef (U)
TrackingNumber (U)
Personnr (FK)
Customer
Personnr
Name
Phone number (U)
E-mail (U)
Password
Newsletter permission
CustomerAddress
Personnr (FK)
Street
HouseNr
PostalCode
City
Misc

3. 3NF

All tables contain no transitive dependencies and are already in 2NF, therefore the schema is in 3NF



Part 3. Creating tables, populating the database, SQL queries, indices

Table creation:

```sql
CREATE TABLE Customer (
  Email VARCHAR(255) NOT NULL,
  Phone_number VARCHAR(15) NOT NULL,
  `Password` VARCHAR(255) NOT NULL,
  Newsletter_permission BOOL NOT NULL,
  `Name` VARCHAR(255) NOT NULL,
  Personnr VARCHAR(12) NOT NULL,
  PRIMARY KEY (Personnr),
  UNIQUE (Email),
  UNIQUE (Phone_number)
);
CREATE TABLE CustomerAddress (
  Street VARCHAR(255) NOT NULL,
  HouseNr INT NOT NULL,
  PostalCode VARCHAR(20) NOT NULL,
  City VARCHAR(255) NOT NULL,
  Misc VARCHAR(255) NOT NULL,
  Personnr VARCHAR(12) NOT NULL,
  PRIMARY KEY (Personnr),
  FOREIGN KEY (Personnr) REFERENCES Customer(Personnr)
);
CREATE TABLE `Order` (
  OrderID INT NOT NULL,
  OrderDate DATE NOT NULL,
  OrderStatus VARCHAR(255) NOT NULL,
  LastChangeDate DATE NOT NULL,
  PaymentRef INT NOT NULL,
  TrackingNumber INT NOT NULL,
  Personnr VARCHAR(12) NOT NULL,
  PRIMARY KEY (OrderID),
  FOREIGN KEY (Personnr) REFERENCES Customer(Personnr),
  UNIQUE (PaymentRef),
  UNIQUE (TrackingNumber)
);
CREATE TABLE ProductQuantity (
  Quantity INT NOT NULL,
  Price DECIMAL(10, 2) NOT NULL,
  OrderID INT NOT NULL,
  ProductID INT NOT NULL,
  PRIMARY KEY (OrderID, ProductID),
  FOREIGN KEY (OrderID) REFERENCES `Order`(OrderID),
  FOREIGN KEY (ProductID) REFERENCES Product(ProductID)
);
```

```sql
CREATE TABLE Product (
  ProductDesc TEXT NOT NULL,
  ProductTitle VARCHAR(255) NOT NULL,
  VATPercentage DECIMAL(5, 2) NOT NULL,
  StockQuantity INT NOT NULL,
  DiscountPercentage DECIMAL(5, 2) NOT NULL,
  DepartmentID INT NOT NULL,
  ProductID INT NOT NULL,
  RetailPriceNoVAT DECIMAL(10, 2) NOT NULL,
  WebLink VARCHAR(255) NOT NULL,
  Featured VARCHAR(1) NOT NULL,
  PRIMARY KEY (ProductID),
  FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID)
);

CREATE TABLE Related (
  Product1ID INT NOT NULL,
  Product2ID INT NOT NULL,
  Keyword VARCHAR(255) NOT NULL,
  PRIMARY KEY (Product1ID, Product2ID),
  FOREIGN KEY (Product1ID) REFERENCES Product(ProductID),
  FOREIGN KEY (Product2ID) REFERENCES Product(ProductID)
);
CREATE TABLE Review (
  ReviewID INT NOT NULL,
  ReviewText TEXT NOT NULL,
  Rating INT NOT NULL,
  Date DATE NOT NULL,
  Personnr VARCHAR(12) NOT NULL,
  ProductID INT NOT NULL,
  PRIMARY KEY (ReviewID),
  FOREIGN KEY (Personnr) REFERENCES Customer(Personnr),
  FOREIGN KEY (ProductID) REFERENCES Product(ProductID)
);

CREATE TABLE Homepage (
  HomepageID INT NOT NULL,
  HomepageText TEXT NOT NULL,
  PRIMARY KEY (HomepageID)
);

CREATE TABLE Department (
  DepartmentID INT NOT NULL,
  Title VARCHAR(255) NOT NULL,
  ShortDescr TEXT NOT NULL,
  Descr TEXT NOT NULL,
  HomepageID INT NOT NULL,
  ParentDepartmentID INT,
  PRIMARY KEY (DepartmentID),
  FOREIGN KEY (HomepageID) REFERENCES Homepage(HomepageID),
  FOREIGN KEY (ParentDepartmentID) REFERENCES Department(DepartmentID)
```

```
);
```

Table population:

```
INSERT INTO Department (DepartmentID, Title, ShortDescr, Descr, HomepageID,
ParentDepartmentID)
VALUES
  (1, 'Electronics', 'Electronics', 'Electronics Department', 1, NULL),
  (2, 'Clothing', 'Clothing', 'Clothing Department', 1, NULL),
  (3, 'Computers', 'Computers', 'Computers Department', 1, 1),
  (4, 'Home Appliances', 'Home Appliances', 'Home Appliances Department', 1, 1),
  (5, 'Men''s Clothing', 'Men''s Clothing', 'Men''s Clothing Department', 1, 2),
  (6, 'Women''s Clothing', 'Women''s Clothing', 'Women''s Clothing Department', 1, 2),
  (7, 'Televisions', 'TVs', 'TV Department', 1, 1),
  (8, 'Children''s Clothing', 'Children''s Clothing', 'Children''s Clothing Department',
1, 2);


INSERT INTO Product (ProductID, ProductDesc, ProductTitle, VATPercentage, StockQuantity,
DiscountPercentage, DepartmentID, RetailPriceNoVAT, WebLink, Featured)
VALUES
  (1, 'Smartphone with advanced features', 'SmartTech X1', 20.00, 50, 5.00, 1, 450.00,
'https://example.com/product/1', 1),
  (2, 'Noise-Canceling Wireless Headphones', 'QuietBeats Pro', 20.00, 30, 10.00, 1,
200.00, 'https://example.com/product/2' , 1),
  (3, 'Men''s Casual Shirt', 'CasualChic Men''s Shirt', 20.00, 100, 15.00, 2, 35.00,
'https://example.com/product/3' , 0 ),
  (4, 'Women''s Summer Dress', 'SummerBloom Women''s Dress', 20.00, 80, 10.00, 2, 45.00,
'https://example.com/product/4', 0),
  (5, 'High-Performance Gaming PC', 'GameMaster Pro', 20.00, 20, 8.00, 3, 1200.00,
'https://example.com/product/5', 0),
  (6, 'Ultra-Slim Laptop', 'SlimBook X2', 20.00, 50, 12.00, 3, 800.00,
'https://example.com/product/6', 1),
  (7, 'Smart Refrigerator with AI Features', 'SmartFrost 3000', 20.00, 15, 5.00, 4,
900.00, 'https://example.com/product/7', 0),
  (8, 'Energy-Efficient Washing Machine', 'EcoWash Pro', 20.00, 25, 8.00, 4, 600.00,
'https://example.com/product/8', 0),
  (9, 'Men''s Classic Suit', 'ClassicStyle Men''s Suit', 20.00, 40, 10.00, 5, 150.00,
'https://example.com/product/9' , 0),
  (10, 'Men''s Casual Jeans', 'ComfortFit Men''s Jeans', 20.00, 60, 12.00, 5, 40.00,
'https://example.com/product/10', 0),
  (11, 'Elegant Women''s Dress', 'Elegance in Silk Dress', 20.00, 35, 15.00, 6, 85.00,
'https://example.com/product/11', 0),
  (12, 'Women''s Denim Jacket', 'Denim Delight Jacket', 20.00, 55, 8.00, 6, 55.00,
'https://example.com/product/12', 0);

INSERT INTO Related (Product1ID, Product2ID, Keyword)
VALUES
  (1, 2, 'Innovative'),
  (1, 6, 'Minimalistic'),
  (1, 6, 'Latest tech');
```

```sql
INSERT INTO Customer (Personnr, Email, Phone_number, Password, Newsletter_permission,
Name)
VALUES
  (012345, 'user1@example.com', '1234567890', 'password1', 1, 'User 1'),
  (012346, 'user2@example.com', '9876543210', 'password2', 0, 'User 2'),
  (012347, 'user3@example.com', '1112233445', 'password3', 1, 'User 3'),
  (012348, 'user4@example.com', '9988776655', 'password4', 1, 'User 4'),
  (012349, 'user5@example.com', '2233445566', 'password5', 0, 'User 5');

INSERT INTO Review (ReviewID, ReviewText, Rating, Date, Personnr, ProductID)
VALUES
  (1, 'Great smartphone!', 4, '2023-09-20', 1, 1),
  (2, 'Excellent laptop for work.', 4, '2023-09-21', 1, 2),
  (3, 'Highly recommend this dress.', 5, '2023-09-22', 2, 3),
  (4, 'Best headphones I ever had!', 5, '2023-09-23', 2, 2),
  (5, 'Comfortable and stylish.', 4, '2023-09-24', 3, 4),
  (6, 'OK performance.', 2, '2023-09-25', 1, 1);

INSERT INTO `Order` (OrderID, OrderDate, OrderStatus, LastChangeDate, PaymentRef,
TrackingNumber, Personnr)
VALUES
  (1, '2023-09-23', 'Shipped', '2023-09-24', 12345, 54321, 1);

INSERT INTO Homepage (HomepageID, HomepageText)
VALUES
  (1, 'Check out our featured products!');
```

SQL Queries:

```sql
-- 1.
SELECT HomepageText
FROM Homepage;

-- 2.
SELECT D.DepartmentID, D.Title, D.ShortDescr
FROM Department D
WHERE D.ParentDepartmentID IS NULL;

-- 3.
SELECT *
FROM Product
WHERE Featured = '1';

-- 4.
SELECT * FROM Related
WHERE Product1ID = 1 OR Product2ID = 1;

-- 5.
SELECT P.ProductTitle, P.ProductDesc, P.RetailPriceNoVAT, AVG(R.Rating) AS AvgRating
FROM Product P
LEFT JOIN Review R ON P.ProductID = R.ProductID
WHERE P.DepartmentID = 2
GROUP BY P.ProductTitle, P.ProductDesc, P.RetailPriceNoVAT;

-- 6.
SELECT P.ProductTitle, P.DiscountPercentage
FROM Product P
WHERE P.DiscountPercentage <> 0
ORDER BY P.DiscountPercentage DESC;
CREATE INDEX idx_discount_percentage ON Product (DiscountPercentage);
```

Before:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	P	NULL	ALL	NULL	NULL	NULL	NULL	12	90.00	Using where; Using filesort

After:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	P	NULL	ALL	idx_discount_percentage	NULL	NULL	NULL	12	100.00	Using where; Using filesort

Part 4. Python algorithm for interaction with the DB

```
## Task 1 and 2

import pymysql
from sshtunnel import SSHTunnelForwarder
tunnel = SSHTunnelForwarder(
 ('fries.it.uu.se', 22),
 ssh_username = 'saja2454',
 ssh_password = 'S.j.A.1377',
 remote_bind_address=('127.0.0.1', 3306)
 )
tunnel.start()
conn = pymysql.connect(
 host='127.0.0.1',
 user='ht23_1_group_39',
 password='pasSWd_39',
 port=tunnel.local_bind_port,
 db = 'ht23_1_project_group_39'
 )

cursor = conn.cursor()


 # Task 1: List products or child departments of a given department

def list_products_or_child_departments(department_id):
     query = "SELECT Title, DepartmentID FROM Department WHERE
ParentDepartmentID = %s"
     cursor.execute(query, (department_id,))
     child_departments = cursor.fetchall()

     if child_departments:
         print("Child Departments:")
         for department in child_departments:
             print(f"ID: {department[1]}, Title: {department[0]}")
     else:
         query = "SELECT P.ProductID, P.ProductTitle, P.RetailPriceNoVAT,
(P.RetailPriceNoVAT - (P.RetailPriceNoVAT * P.DiscountPercentage / 100)) AS
DiscountedPrice FROM Product P WHERE DepartmentID = %s"
         cursor.execute(query, (department_id,))
         products = cursor.fetchall()
```

```
        if products:
            print("Products in the Department:")
            for product in products:
                print(f"ID: {product[0]}, Title: {product[1]}, Discounted
Price: ${product[3]:.2f}")
        else:
            print("No child departments or products found for the given
department ID.")




# Task 2: Change the discount percentage of a product
def change_discount(product_id):

    query = "SELECT ProductTitle, DiscountPercentage FROM Product WHERE
ProductID = %s"
    cursor.execute(query, (product_id,))
    result = cursor.fetchone()

    if result:
        current_discount = result[1]
        product_name = result[0]

        print(f"Current Discount for Product {product_name} (ID:
{product_id}): {current_discount}%")

        new_discount = input("Enter the new discount percentage: ")
        query = "UPDATE Product SET DiscountPercentage = %s WHERE ProductID
= %s"
        cursor.execute(query, (new_discount, product_id))
        conn.commit()
        print(f"Discount for Product {product_name} (ID: {product_id})
updated to {new_discount}%")

    else:
        print(f"No product found with Product ID {product_id}.")


 # Take user input for the tasks
while True:
    print("\nChoose a task:")
    print("1. List products or child departments of a department")
    print("2. Change the discount of a product")
```

```
    print("0. Exit")
    choice = input("Enter the task number (0/1/2): ")

    if choice == "1":
        department_id = input("Enter the department ID: ")
        list_products_or_child_departments(department_id)
    elif choice == "2":
        product_id = input("Enter the product ID: ")
        change_discount(product_id)
    elif choice == "0":
        break
    else:
        print("Invalid choice. Please enter a valid task number.")

cursor.close()
conn.close()
```

This Python program is designed to interact with the product database. It allows users to perform two main tasks:

1. List products or child departments of a given department,
2. Change the discount percentage of a product.

The program connects to a MySQL database using the pymysql, retrieves data, and updates the database as necessary.

Instructions:

Upon running the program, you will be presented with a menu to choose a task. You can enter 0, 1, or 2 to select a task.

Task 1 - List Products or Child Departments:

When choosing Task 1, you will be prompted to enter a department ID. You can input the department ID to list its products or child departments.

Task 2 - Change Discount Percentage:

If you select Task 2, you will be asked to input a product ID and the new discount percentage. The program will display the current discount and product name before allowing you to update the discount percentage.

Sample Inputs for Each Task:

Task 1: List Products or Child Departments

```
Choose a task:
1. List products or child departments of a department
2. Change the discount of a product
0. Exit
Enter the task number (0/1/2): 1
Enter the department ID: 2
Child Departments:
ID: 5, Title: Men's Clothing
ID: 6, Title: Women's Clothing

Choose a task:
1. List products or child departments of a department
2. Change the discount of a product
0. Exit
Enter the task number (0/1/2): 1
Enter the department ID: 5
Products in the Department:
ID: 9, Title: ClassicStyle Men's Suit, Discounted Price: $135.00
ID: 10, Title: ComfortFit Men's Jeans, Discounted Price: $35.20
```

Task 2: Change the discount percentage of a product

```
Choose a task:
1. List products or child departments of a department
2. Change the discount of a product
0. Exit
Enter the task number (0/1/2): 2
Enter the product ID: 3
Current Discount for Product CasualChic Men's Shirt (ID: 3): 15.00%
Enter the new discount percentage: 35
Discount for Product CasualChic Men's Shirt (ID: 3) updated to 35%
```