Midterm Project Report

Ecommerce Database System

Database Management System 2

Prepared by
Zhanar Mazhit, ID: 210103161
Nurbol Akhmetov, ID: 210101084

24.04.2023

Introduction To the System

The sample for our ecommerce system is Flip.kz. Flip.kz is one of the first online stores in Kazakhstan that sells books, household items, pet products, stationery, clothing, etc. The purpose of our database is to provide efficient storage of data and facilitate the work for data analysis. The database consists of many tables and by working together they create a good environment for future work. Along with that, there are five queries that can be implemented to solve some business problems.

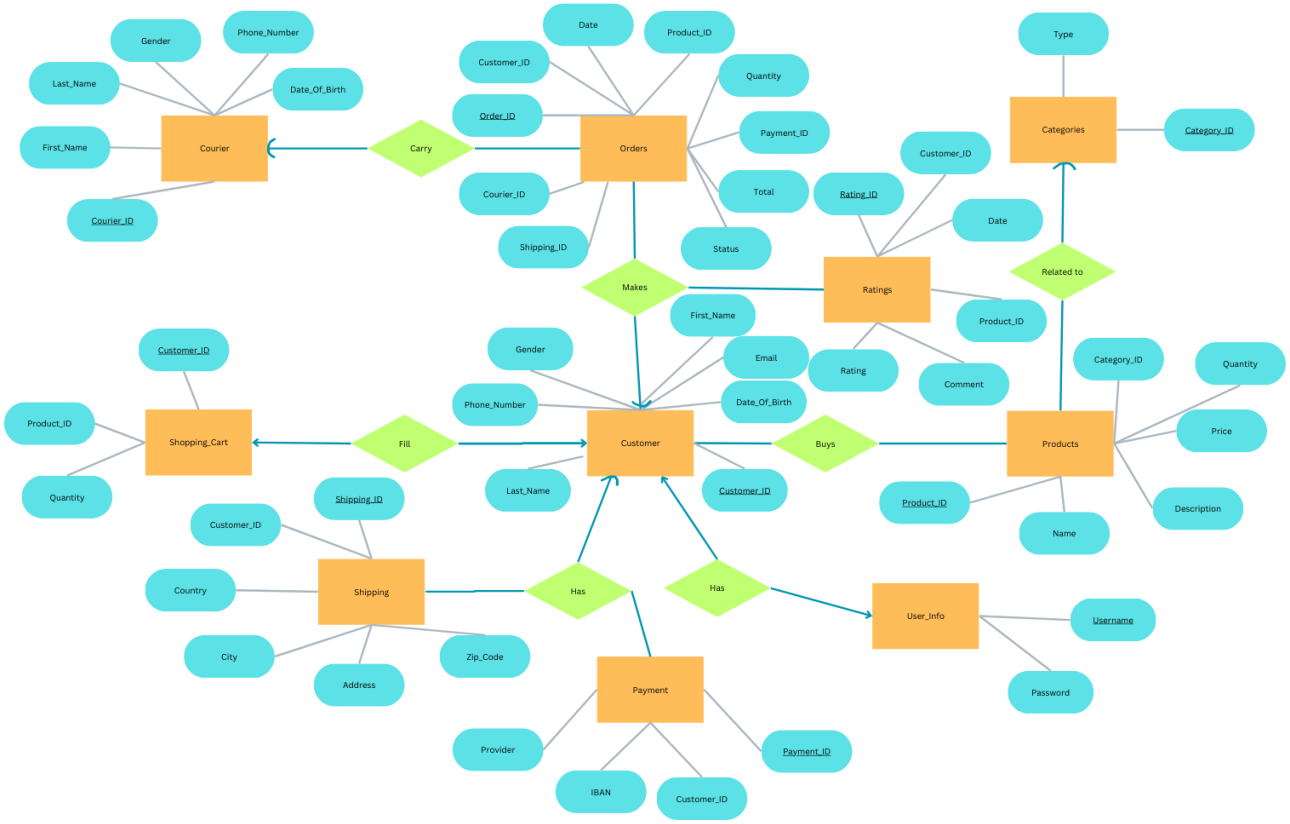
There are 10 entities in our database. They are: Customers, User_Info, Payment, Shopping_Cart, Orders, Shipping, Rating, Product, Category, Courier.

Customer table's attributes: Customer_ID (PK), First_Name, Last_Name, Birth_Date, Email, Gender, Phone_Number.
User_Info table's attributes: Username (PK), Password.
Payment table's attributes: Payment_ID (PK), Customer_ID, IBAN, Provider.
Shopping_Cart table's attributes: Customer_ID (PK), Product_ID, Quantity.
Orders table's attributes: Order_ID (PK), Customer_ID, Date, Product_ID, Quantity, Payment_ID, Status, Shipping_ID, Courier_ID.
Shipping table's attributes: Shipping_ID (PK), Customer_ID, City, Address, Zip_Code.
Rating table's attributes: Rating_ID (PK), Customer_ID, Product_ID, Comment_Text, Rating, Rating_Date.
Product table's attributes: Product_ID (PK), Name, Description, Price, Category_ID, Quantity.
Category table's attributes: Category_ID (PK), Type.
Courier table's attributes: Courier_ID (PK), First_Name, Last_Name, Gender, Phone_Number, Date_Of_Birth.

The business process can be described as this: customer registers at the platform, where his login credentials, such as username and password are stored, then he can explore the store and add the products in the shopping cart. Each product, in its turn, is related to a particular category. After deciding what products the customer will purchase, the process of ordering will start. Firstly, the customer should enter

payment and shipping information, which will be used while making an order. Then a bank should approve the transaction and the courier will be selected. The order statuses will change depending on the part of the process. There are four stages, such as pending, processing, shipped, delivered. In cases where the customer returns the product the status will be ‘Returned’ or when the customer cancels the order, the status will be ‘Canceled’. Lastly, customers can leave ratings along with comments.

ER Diagram



Relationships:

One-to-one:

- Each customer can have one login information and each login information can be assigned to only one person.
- Each customer can have one shopping cart and each shopping cart can be assigned to one customer.

Many-to-one:

- Each customer can have one shipping address but each shipping address can be assigned to many customers.
- Each customer can have one payment method but each payment method can be assigned to many customers.
- Many customers can make many orders but each order can be made by one customer.

- Each customer can make many ratings but each rating can be made by one customer.
- Each category can have many products but each product is assigned to one category.
- Many couriers can carry many orders but one order can be carried by one courier.

Many-to-many:

- Many customers can buy many products.

Normalization In the Tables

Every entity follows the 1NF because each table cell has exactly one value and each record is unique, meaning that there are no duplicate rows. 2NF rules are followed, as well, because all non-key attributes are fully functional dependent on the primary key which is defined in each entity as ID. And, finally, all tables are in 3NF because there are no transitive functional dependencies. The non-key values in the tables cannot define other non-key values.
Courier table can be used as an example:

COURIER_ID	FIRST_NAME	LAST_NAME	GENDER	PHONE_NUMBER	BIRTH_DATE	EMAIL
2482410	Wakefield	Longforth	M	733-221-0802	03/28/1997	wlongforth0@reddit.com
2698705	Wilmar	Clemas	M	894-390-3105	02/22/1991	wclemas1@amazon.de
2621742	Nevile	Jedrzej	M	629-271-8352	06/18/1990	njedrzej2@technorati.com
2898478	Billy	Kynan	M	833-254-4812	05/29/1990	bkynan3@google.nl
2433414	Trev	Matyukon	M	893-280-0212	07/10/1993	tmatyukon4@mtv.com
2648720	Kippie	Sinclaire	M	971-299-5309	12/31/1986	ksinclaire5@youku.com
2830158	Hatti	Swains	F	344-470-2819	10/24/1984	hswains6@geocities.com
2468536	Lanette	Jorry	F	826-594-2632	05/19/1992	ljorry7@tripod.com

It is noticeable each table cell has exactly one value. All non-key attributes, such as First_Name, Last_Name, Gender, Phone_Number, Birth_Date rely on the Courier_ID attribute. Lastly, non-key attribute, for instance, Last_Name cannot define other non-key attributes.

Queries Of the Database

1.Here is the GROUP BY procedure. We use it to group up the data based on one or more rows in the table.

Declaration:

```
CREATE OR REPLACE PROCEDURE grouping_up AS
BEGIN
  FOR rec IN (
    SELECT first_name, last_name, gender, birth_date, COUNT(*) AS
cust_num
    FROM CUSTOMER
    GROUP BY first_name, last_name, gender, birth_date
    ORDER BY first_name, last_name, gender, birth_date
  )
  LOOP
    DBMS_OUTPUT.PUT_LINE(
      rec.first_name || ' ' || rec.last_name || ' (' || rec.gender || ') was born ' ||
rec.birth_date ||
      ' has ' || rec.cust_num || ' customers'
    );
  END LOOP;
END;
```

Execution:

```
BEGIN
grouping_up;
END;
```

2. Function called “count_orders” that counts the number of orders that a particular customer made:

Declaration:

```
CREATE OR REPLACE FUNCTION count_orders(p_id IN NUMBER)
RETURN NUMBER IS
```

```
count_orders INT;
BEGIN
SELECT COUNT(*) INTO count_orders
FROM Orders
WHERE Customer_ID = p_id;
RETURN count_orders;
END;
```

Execution:

```
DECLARE
c_id INT := 1499397;
output INT;
BEGIN
output := count_orders(c_id);
dbms_output.put_line(output);
END;
```

3. Here we create the procedure that uses SQL%ROWCOUNT in order to determine the number of affected rows. For example, the row price.

Declaration:

```
CREATE OR REPLACE PROCEDURE row_count
AS
new_price number;
BEGIN
UPDATE PRODUCT
   SET PRICE = 125
   WHERE QUANTITY > 30;
new_price := SQL%ROWCOUNT;
   DBMS_OUTPUT.PUT_LINE(new_price);
END;
```

Execution:

```
BEGIN
row_count;
END;
```

4. User-defined exception that raises when an entered title's length is less than 5 characters:

```
DECLARE
  product_name VARCHAR2(50) := 'Narf';
BEGIN
  IF LENGTH(product_name) < 5 THEN
    RAISE_APPLICATION_ERROR(-20001, 'Product name length must be greater than 5!');
  END IF;
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
```

5. Here we created a trigger for showing the quantity of rows in the table after adding a new row:

Creation of trigger:

```
CREATE OR REPLACE TRIGGER show_row_count
BEFORE INSERT ON Category
FOR EACH ROW
BEGIN
  DBMS_OUTPUT.PUT_LINE('Current row count: ' || SQL%ROWCOUNT);
END;
```

Insertion in the table:

```
INSERT INTO CATEGORY (Category_ID) VALUES (12)
```