SNEAKERS SHOP

DATABASE MANAGEMENT SYSTEMS

By Nuraddinov Alikhan, Yessenbay Galym

DATABASE MANAGEMENT SYSTEMS

PRESENTATION CONTENT

Introduction to the system
ER diagram
Normal forms
Coding part

SNEAKERS SHOP

INVENTORY MANAGEMENT

The first step in this process is to manage the inventory of sneakers available for purchase

WEBSITE DESIGN AND DEVELOPMENT

After the inventory is processed, the seller needs to design and develop an e-commerce website where buyers can view the available sneakers and place orders.

PRODUCT LISTINGS

The retailer needs to create product listings for each sneaker

ORDER MANAGEMENT

The seller must provide a secure payment processing system that allows customers to pay for their orders using a variety of payment methods.

Introduction to the system

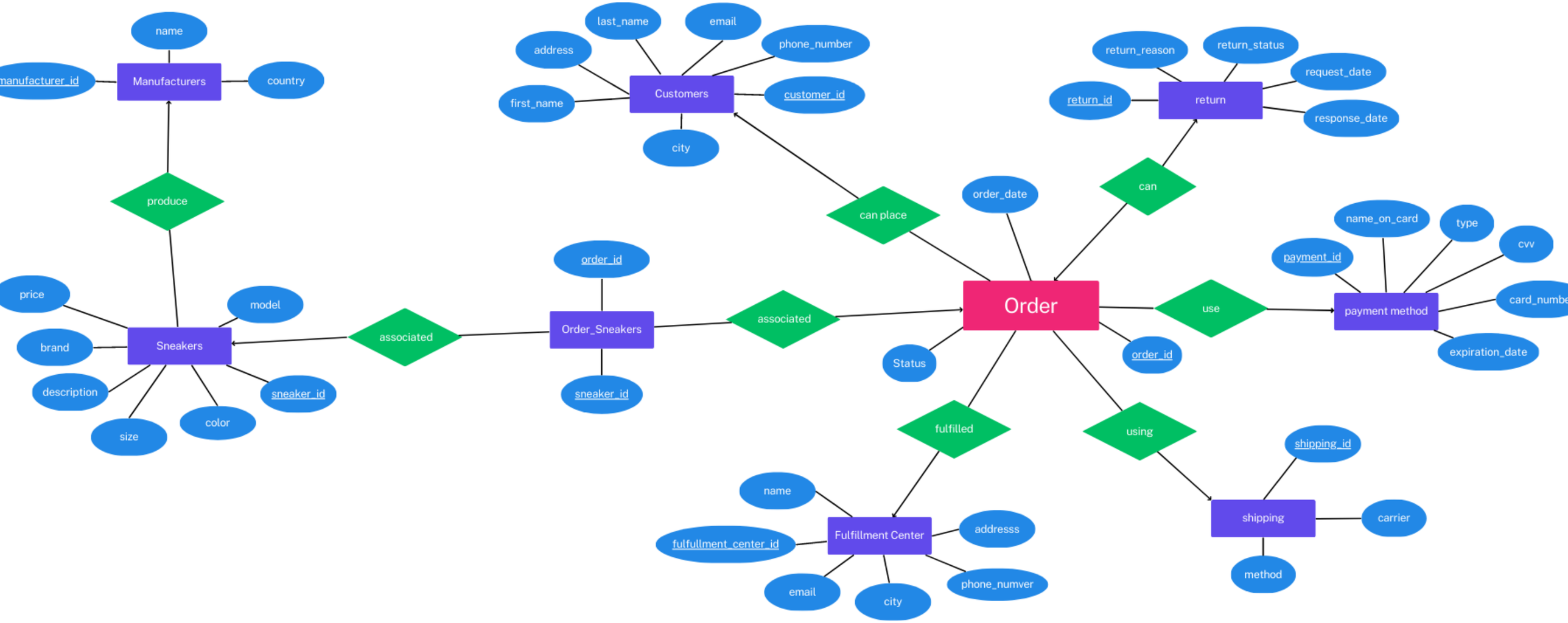
PAYMENT PROCESSING

The seller must provide a secure payment processing system

SHIPPING AND DELIVERY

The seller must fulfill orders and send the sneakers to the specified customer addresses

ER/DIAGRAM


name
manufacturer_id
Manufacturers
country
produce
Sneakers
price
model
brand
description
size
color
sneaker_id
associated
Order_Sneakers
order_id
sneaker_id
associated
last_name
email
address
phone_number
first_name
Customers
customer_id
city
can place
Order
order_date
Status
order_id
return_reason
return_status
request_date
return_id
return
response_date
can
use
name_on_card
type
cvv
payment_id
payment method
card_numbe
expiration_date
fulfilled
using
name
Fulfillment Center
addresss
fulfullment_center_id
email
city
phone_numver
shipping_id
shipping
carrier
method

NORMAL FORMS

THE "MANUFACTURER" TABLE IS IN 3NF

THE "SNEAKERS" TABLE IS IN 3NF

THE "CUSTOMERS" TABLE IS IN 3NF

THE "PAYMENT_METHOD" TABLE IS IN 3NF

THE "ORDER_DETAILS" TABLE IS IN 3NF

THE "RETURN" TABLE IS IN 3NF

THE "FULFILLMENT_CENTER" TABLE IS IN 3NF

THE "SHIPPING" TABLE IS IN 3NF

THE "ORDERS" TABLE IS IN 2NF

CODING PART

- Create tables
- Procedure which does group by information
- Function which counts the number of records
- Procedure which uses SQL%ROWCOUNT
- Add user-defined exception
- trigger that will show the current number of rows in the table

Create tables

```sql
CREATE TABLE Sneakers (
  sneaker_id NUMBER(10) PRIMARY KEY,
  brand VARCHAR2(50) NOT NULL,
  model VARCHAR2(50) NOT NULL,
  sizes NUMBER(10,2) NOT NULL,
  color VARCHAR2(50) NOT NULL,
  price NUMBER(10,2) NOT NULL,
  description VARCHAR2(500),
  manufacturer_id NUMBER(10) NOT NULL,
  CONSTRAINT fk_sneakers_manufacturer FOREIGN KEY (manufacturer_id)
    REFERENCES Manufacturer(manufacturer_id)
);

CREATE TABLE Manufacturer (
  manufacturer_id NUMBER(10) PRIMARY KEY,
  name VARCHAR2(50) NOT NULL,
  country VARCHAR2(50) NOT NULL
);

CREATE TABLE Customers (
  customer_id NUMBER(10) PRIMARY KEY,
  last_name VARCHAR2(50) NOT NULL,
  first_name VARCHAR2(50) NOT NULL,
  email VARCHAR2(50) UNIQUE NOT NULL,
  city VARCHAR2(50) NOT NULL,
  address VARCHAR2(100) NOT NULL,
  phone_number VARCHAR2(20) NOT NULL
);
```

```sql
CREATE TABLE Orders (
  order_id NUMBER(10) PRIMARY KEY,
  order_date DATE NOT NULL,
  status VARCHAR2(50) NOT NULL,
  customer_id NUMBER(10) NOT NULL,
  payment_method_id NUMBER(10),
  shipping_id NUMBER(10),
  fulfillment_center_id NUMBER(10),
  sneakers_id NUMBER(10),
  CONSTRAINT fk_order_customer FOREIGN KEY (customer_id)
    REFERENCES Customers(customer_id),
  CONSTRAINT fk_order_payment FOREIGN KEY (payment_method_id)
    REFERENCES Payment_Method(payment_id),
  CONSTRAINT fk_order_shipping FOREIGN KEY (shipping_id)
    REFERENCES Shipping(shipping_id),
  CONSTRAINT fk_order_fulfillment FOREIGN KEY (fulfillment_center_id)
    REFERENCES Fulfillment_Center(fulfillment_center_id),
  CONSTRAINT fk_order_sneakers FOREIGN KEY (sneakers_id)
    REFERENCES Sneakers(sneaker_id)
);

CREATE TABLE Payment_Method (
  payment_id NUMBER(10) PRIMARY KEY,
  type VARCHAR2(50) NOT NULL,
  name_on_card VARCHAR2(100) NOT NULL,
  card_number VARCHAR2(50) NOT NULL,
  expiration_date DATE NOT NULL,
  cvv VARCHAR2(10) NOT NULL
);
```

```sql
CREATE TABLE Fulfillment_Center (
  fulfillment_center_id NUMBER(10) PRIMARY KEY,
  name VARCHAR2(50) NOT NULL,
  address VARCHAR2(100) NOT NULL,
  phone_number VARCHAR2(20) NOT NULL,
  city VARCHAR2(50) NOT NULL,
  email VARCHAR2(50) NOT NULL
);

CREATE TABLE Return (
  return_id NUMBER(10) PRIMARY KEY,
  return_reason VARCHAR2(100) NOT NULL,
  return_status VARCHAR2(50) NOT NULL,
  request_date DATE NOT NULL,
  response_date DATE,
  refund_amount NUMBER(10,2),
  order_id NUMBER(10) UNIQUE NOT NULL,
  CONSTRAINT fk_order_return FOREIGN KEY (order_id)
    REFERENCES Orders(order_id),
);

CREATE TABLE Order_Details (
  order_id NUMBER(10) NOT NULL,
  sneakers_id NUMBER(10) NOT NULL,
  CONSTRAINT pk_order_details PRIMARY KEY (order_id, sneakers_id),
  CONSTRAINT fk_order_details_order FOREIGN KEY (order_id)
    REFERENCES Orders(order_id),
  CONSTRAINT fk_order_details_sneakers FOREIGN KEY (sneakers_id)
    REFERENCES Sneakers(sneaker_id)
);
```

```sql
CREATE TABLE Order_Details (
  order_id NUMBER(10) NOT NULL,
  sneakers_id NUMBER(10) NOT NULL,
  CONSTRAINT pk_order_details PRIMARY KEY (order_id, sneakers_id),
  CONSTRAINT fk_order_details_order FOREIGN KEY (order_id)
    REFERENCES Orders(order_id),
  CONSTRAINT fk_order_details_sneakers FOREIGN KEY (sneakers_id)
    REFERENCES Sneakers(sneaker_id)
);
```

Procedure which does group by information

```sql
CREATE OR REPLACE PROCEDURE group_sneakers_by_brand
AS
BEGIN
  FOR brand_rec IN (SELECT brand, COUNT(*) AS total_count FROM Sneakers GROUP BY brand)
  LOOP
    DBMS_OUTPUT.PUT_LINE('Brand: ' || brand_rec.brand || ', Total Count: ' || brand_rec.total_count);
  END LOOP;
END;
```

```
Procedure created.


0.04 seconds
```

```sql
BEGIN
  group_sneakers_by_brand;
END;
```

```
Brand: Adidas, Total Count: 4
Brand: Puma, Total Count: 3
Brand: Gucci, Total Count: 2
Brand: Nike, Total Count: 6
Brand: New Balance, Total Count: 3
Brand: Timberland, Total Count: 1
Brand: Jordan, Total Count: 3

Statement processed.


0.03 seconds
```

Function which counts the number of records

```sql
CREATE OR REPLACE FUNCTION count_records(table_name IN VARCHAR2) RETURN NUMBER IS
  num_records NUMBER;
BEGIN
  EXECUTE IMMEDIATE 'SELECT COUNT(*) FROM ' || table_name INTO num_records;
  RETURN num_records;
END;
```

```
Function created.


0.05 seconds
```

```sql
SELECT count_records('Sneakers') as sneaker_count FROM dual;
SELECT count_records('Manufacturer') as manufacturer_count FROM dual;
SELECT count_records('Customers') as customer_count FROM dual;
SELECT count_records('Orders') as order_count FROM dual;
SELECT count_records('Payment_Method') as payment_method_count FROM dual;
SELECT count_records('Shipping') as shipping_count FROM dual;
SELECT count_records('Fulfillment_Center') as fulfillment_center_count FROM dual;
SELECT count_records('Return') as return_count FROM dual;
```

CUSTOMER_COUNT
20

1 rows returned in 0.11 seconds Download

PAYMENT_METHOD_COUNT
20

1 rows returned in 0.12 seconds Download

Procedure which uses SQL%ROWCOUNT

```sql
CREATE OR REPLACE PROCEDURE update_sneakers_price(
  in_price NUMBER,
  in_sneaker_id NUMBER
)
IS
BEGIN
  UPDATE sneakers
  SET price = in_price
  WHERE sneaker_id = in_sneaker_id;

  DBMS_OUTPUT.PUT_LINE(SQL%ROWCOUNT || ' row(s) updated.');

  COMMIT;
END;
```

```
Procedure created.

0.01 seconds
```

```sql
DECLARE
  v_sneaker_id NUMBER := 1;
  v_new_price NUMBER := 150.00;
BEGIN
  update_sneakers_price(in_price => v_new_price, in_sneaker_id => v_sneaker_id);
END;
```

```
1 row(s) updated.

Statement processed.

0.02 seconds
```

Add user-defined exception

```
CREATE OR REPLACE TRIGGER title_length_check
BEFORE INSERT ON Sneakers
FOR EACH ROW
DECLARE
  title_length EXCEPTION;
BEGIN
  IF LENGTH(:new.model) < 5 THEN
    RAISE title_length;
  END IF;
EXCEPTION
  WHEN title_length THEN
    RAISE_APPLICATION_ERROR(-20001, 'Title must be at least 5 characters long');
END;
```

```
Trigger created.


0.06 seconds
```

```
INSERT INTO Sneakers (sneaker_id, brand, model, sizes, color, price, description, manufacturer_
VALUES (1, 'Nike', 'Air', 9.5, 'Red', 100, 'Comfortable shoes', 1);
```

```
ORA-20001: Title must be at least 5 characters long
ORA-06512: at "WKSP_GALYM.TITLE_LENGTH_CHECK", line 9
ORA-04088: error during execution of trigger 'WKSP_GALYM.TITLE_LENGTH_CHECK'
ORA-06512: at "SYS.DBMS_SQL", line 1721
```

```
INSERT INTO Sneakers (sneaker_id, brand, model, sizes, color, price, description, manufacturer_id)
VALUES (21, 'Nike', 'Jordan', 9.5, 'Red', 100, 'Comfortable shoes', 1);
```

```
Current number of rows in Customers table: 20

1 row(s) inserted.
```

Trigger that will show the current number of rows in the table

```sql
CREATE OR REPLACE TRIGGER trg_show_rows_count
BEFORE INSERT ON SNEAKERS
DECLARE
  l_count NUMBER;
BEGIN
  SELECT COUNT(*) INTO l_count FROM Customers;
  DBMS_OUTPUT.PUT_LINE('Current number of rows in Customers table: ' || l_count);
END;
```

```
Trigger created.


0.06 seconds
```

```sql
INSERT INTO Sneakers (sneaker_id, brand, model, sizes, color, price, description, manufacturer_id)
VALUES (21, 'Nike', 'Jordan', 9.5, 'Red', 100, 'Comfortable shoes', 1);
```

```
Current number of rows in Customers table: 21

1 row(s) inserted.


0.01 seconds
```

THANKS FOR ATTENTION