Introduction to Database Systems Bike Store Management System Project

Açelyanur Şen

December 2021

1 Introduction

In this project, I tried to create a database system for a bike dealer who owns three bike shops. With this system, they can keep track of any information about the bikes, customers and their contact information, details about the staff that works in these shops, and the sales they have done. It is useful for seeing the details about the sales since the system also records the order date, shipping date, and the current status of the orders. So, if any problem occurs with the shipment or the sale in general, it is easy to find every detail about it.

2 Database

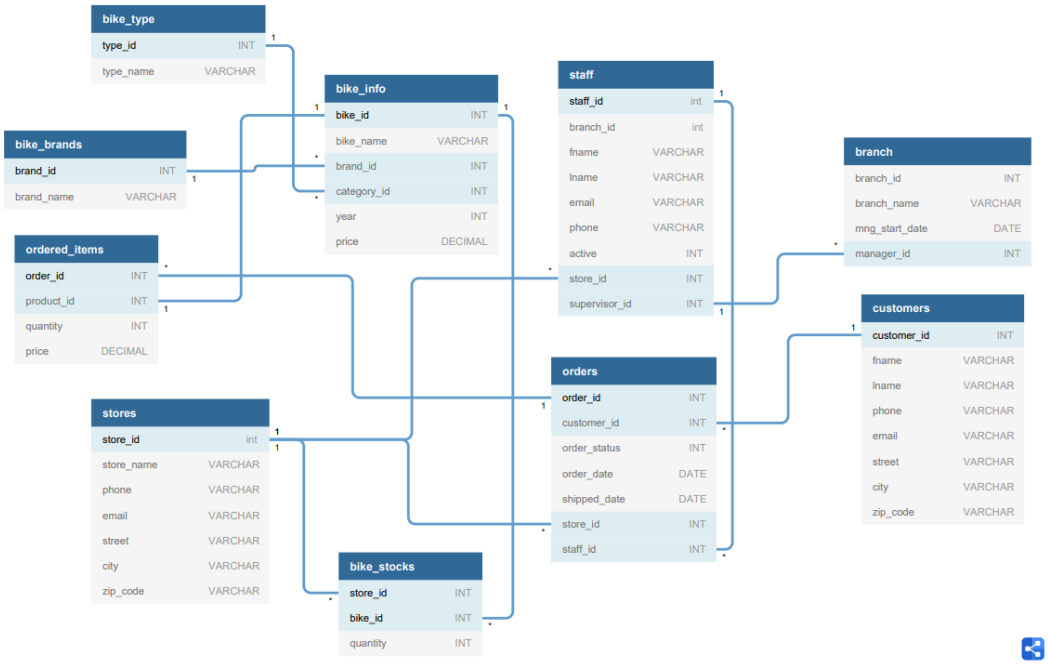

Figure 1: Database diagram with 10 tables and connections between them

As seen in Figure 1 above, the database consist of 10 tables which records information about the bikes, customers, stores and their stocks, orders and ordered items, staff and the branches they work.

2.1 Stores table

Creating the table "stores"

```
CREATE TABLE stores (
store_id INT PRIMARY KEY,
store_name VARCHAR (50) NOT NULL,
phone VARCHAR (25),
email VARCHAR (50),
street VARCHAR (50),
city VARCHAR (50),
zip_code VARCHAR (5)
);
```

2.2 Staff table

Creating the table "staff"

```
CREATE TABLE staff (
staff_id INT PRIMARY KEY,
branch_id INT,
fname VARCHAR (50) NOT NULL,
lname VARCHAR (50) NOT NULL,
email VARCHAR (50) NOT NULL UNIQUE,
phone VARCHAR (25),
active INT NOT NULL,
store_id INT NOT NULL,
supervisor_id INT,
FOREIGN KEY (store_id) REFERENCES stores (store_id)
);
```

Staff table also has a property named "store id" to record which store an employee works at. So, I made "store id" a foreign key linking these two tables.

2.3 Branch table

```
CREATE TABLE branch(
    branch_id INT,
    branch_name VARCHAR(40),
    mng_start_date DATE NOT NULL,
    manager_id INT,
    FOREIGN KEY (manager_id) REFERENCES staff (supervisor_id)

);
```

Branch table will record the departments that the employees work. It will record the starting day of the managers of these departments as well. I made "manager id" a foreign key linking **Branch** and **Staff** table.

2.4 Bike Type table

```
CREATE TABLE bike_type (
type_id INT PRIMARY KEY,
type_name VARCHAR (50) NOT NULL
);
```

Storing the types of the bikes. "Electric bikes" etc.

2.5 Bike Brand table

```
CREATE TABLE bike_brands (
brand_id INT PRIMARY KEY,
brand_name VARCHAR (50) NOT NULL
);
```

For the brands of the bikes.

2.6 Bike Information table

```
CREATE TABLE bike_info (
bike_id INT PRIMARY KEY,
bike_name VARCHAR (50) NOT NULL,
brand_id INT NOT NULL,
category_id INT NOT NULL,
year INT NOT NULL,
price DECIMAL (5, 2) NOT NULL,
FOREIGN KEY (category_id) REFERENCES bike_type (type_id),
FOREIGN KEY (brand_id) REFERENCES bike_brands (brand_id)
);
```

With two foreign keys, I linked this table with Bike Type table.

2.7 Customers table

```
    CREATE TABLE customers (
customer_id INT PRIMARY KEY,
fname VARCHAR (50) NOT NULL,
lname VARCHAR (50) NOT NULL,
phone VARCHAR (25),
email VARCHAR (50) NOT NULL,
street VARCHAR (50),
city VARCHAR (50),
zip_code VARCHAR(50)
);
```

Customer table will keep contact details about the customers.

2.8 Orders table

```
    CREATE TABLE orders (
order_id INT PRIMARY KEY,
customer_id INT,
order_status INT NOT NULL,
-- Order status: 1 = Awaiting approval 2 = Processing 3 = Canceled
4 = Completed
order_date DATE NOT NULL,
shipped_date DATE,
store_id INT NOT NULL,
staff_id INT NOT NULL,
FOREIGN KEY (customer_id) REFERENCES customers (customer_id)
FOREIGN KEY (store_id) REFERENCES stores (store_id) ,
FOREIGN KEY (staff_id) REFERENCES staff (staff_id)
);
```

Orders table stores customer id as well so, it is a connection point between this table and customers. I also keep track of which staff member is responsible with the order. Hence, there's a connection between **Staff** table as well. To keep track of the stock in the shops, there's also shop id in this table too.

2.9 Ordered Items table

```
    CREATE TABLE ordered_items(
order_id INT PRIMARY KEY,
product_id INT NOT NULL,
quantity INT NOT NULL,
price DECIMAL (10, 2) NOT NULL,
FOREIGN KEY (order_id) REFERENCES orders (order_id),
FOREIGN KEY (product_id) REFERENCES bike_info (bike_id)
);
```

Details about which item is ordered will be kept here. It links to the **Bike Information** and **Orders** table.

2.10 Stock table

```
    CREATE TABLE bike_stocks (
store_id INT,
bike_id INT,
quantity INT,
PRIMARY KEY (store_id, bike_id),
FOREIGN KEY (store_id) REFERENCES stores (store_id),
FOREIGN KEY (bike_id) REFERENCES bike_info (bike_id)
        ON DELETE CASCADE ON UPDATE CASCADE
);
```

And lastly, we have stock table to see how many bikes the shops have. I added "ON DELETE CASCADE ON UPDATE CASCADE" so when the stock is updated the other keys' references would change accordingly.

3 Insertions

3.1 Shops

```
    --shops
INSERT INTO stores(store_id, store_name, phone, email, street, city,
zip_code)
VALUES
      (100,'HappyBike','+48 504 982 043','happybikes@bikes.shop',
      'Mogilska 51',
      'Krakow',31545),
      (110,'Silver Board','+48 731 609 774','silver_board@bikes.shop',
      'Olszańska 11','Krakow',31513),
      (120,'TSL - Rowery Elektryczne FAT BIKE','+48 575 847 797',
      'fattybikes@bikes.shop','Kosynierów 5', 'Krakow',31527);

      SELECT * from stores;
```

Inserting the three shop information.

store_id	store_name	phone	email	street	city
100	HappyBike	+48 504 982 043	happybikes@bikes.shop	Mogilska 51	Krakow
110	Silver Board	+48 731 609 774	silver_board@bikes.s...	Olszańska 11	Krakow
120	TSL - Rowery Elektryczne FAT BIKE	+48 575 847 797	fattybikes@bikes.shop	Kosynierów 5	Krakow

Figure 2: Table after selection

3.2 Customers

Example insert:

```
  INSERT INTO customers(customer_id,fname, lname, phone, email,
  street, city, zip_code) VALUES
 (666,'Alperen','Duran',NULL,'nazillili@gmail.com',
 'Stanislawa Skarzyńskiego 5','Krakow',31866);
```

Select statement:

```
    SELECT customer_id, fname, email FROM customers
    ORDER BY customer_id;
```

customer_id	fname	email
43	Aleyna	asureler@gmail.com
44	Büşra	busraakdere@gmail.com
55	Açelya	acelyasen@gmail.com
109	Thad	thad.castro@msn.com
110	Tena	tena.huber@gmail.com
123	Tajuana	tajuana.rollins@msn.com
666	Alperen	nazillili@gmail.com
1035	Dori	dori.alvarez@outlook.com
1250	Eden	edenfromparadise@aol.com

Figure 3: Table after selection

3.3 Bikes

Example insert:

```
    --bikes
INSERT INTO bike_info(bike_id, bike_name, brand_id, category_id,
year, price) VALUES
    (11,'Surly Straggler 650b',8,4,2016,1680.99);
```

Select statement:

```
SELECT bike_name, year,price FROM bike_info
ORDER BY brand_id;
```

Bike Name	year	price
Electra Amsterdam Original 3i	2017	659.99
Electra Townie Original 21D	2006	239.99
Electra Cruiser 1 (24-Inch)	2016	269.99
Sun Bicycles Cruz 7	2017	416.99
Sun Bicycles Atlas X-Type	2017	416.99
Surly Straggler 650b	2016	1680.99
Trek CrossRip+	2018	4499.99

Figure 4: Table after select statement

Select statement to see bikes which cost less than 500 bucks:

```
SELECT bike_name, price FROM bike_info
WHERE price < 500;
```

bike_name	price
Sun Bicycles Cruz 7	416.99
Sun Bicycles Atlas X-Type	416.99
Electra Townie Original 21D	239.99
Electra Cruiser 1 (24-Inch)	269.99

Figure 5: Table after select statement

3.4 Bike Types and Stock

Example insert from both tables:

```
--bike type
INSERT INTO bike_type(type_id,type_name)
VALUES(5,'Electric Bikes');

---stock
INSERT INTO bike_stocks(store_id, bike_id, quantity)
VALUES(100,186,6);
```

3.5 Staff

Example insert statement:

```
INSERT INTO staff(staff_id, branch_id, fname, lname, email,
phone, active, store_id, supervisor_id) VALUES(4,2,'Angela'
,'Merkel','bye.merkel@bike.shop','+49 333 555 2222',1,3,6);
```

Select statement:

```
SELECT staff_id, fname, supervisor_id FROM staff
ORDER BY supervisor_id;
```

staff_id	fname	supervisor_id
2	Fahd	Null
6	Bernie	Null
1	Fab	2
8	Patrick	2

Figure 6: Part ot the table after select statement

Another select statement:

```
--who number 6 supervises
SELECT fname FROM staff
WHERE supervisor_id = 6;
```

fname
Coma
Angela
Ted
Abi

Figure 7: Employees under the manager with ID number 6

3.6 Branches and Brands

Example insert statements for both tables:

```
--branches
INSERT INTO branch(branch_id, branch_name, mng_start_date,
manager_id)
VALUES(1,'Sales','2017-06-03',2);

--brands
INSERT INTO bike_brands(brand_id,brand_name)
VALUES(100,'Electra');
```

3.7 Orders

Insert statement:

```
INSERT INTO orders(order_id, customer_id, order_status,
order_date, shipped_date, store_id, staff_id)
VALUES(500,1250,4,'2020-01-15','2020-01-18',2,2);
```

Select statement:

```
SELECT order_id, customer_id, order_date FROM orders
ORDER BY order_date;
```

order_id	customer_id	order_date
405	666	2018-06-17
300	44	2018-07-01
500	1250	2020-01-15
501	666	2020-01-16
400	43	2020-04-29

Figure 8: Part ot the table after select statement

3.8 Ordered Items

Insert statement:

```
INSERT INTO ordered_items(order_id, product_id, quantity,
price)
VALUES(424,12,2,549.99);
```

Select statement to define order status:

```
SELECT order_date, order_id, shipped_date, CASE order_status
WHEN 1 THEN 'Awaiting'
WHEN 2 THEN 'Processing' WHEN 3 THEN 'Canceled'
WHEN 4 THEN 'Completed' END as status
FROM orders
ORDER BY order_date;
```

order_date	order_id	shipped_date	status
2018-06-17	405	2018-06-25	Completed
2018-07-01	300	2018-07-10	Completed
2020-01-15	500	2020-01-18	Completed
2020-01-16	501	2020-01-17	Completed
2020-04-29	400	NULL	Canceled
2020-04-30	423	NULL	Canceled

Figure 9: Part ot the table after select statement

4 Some example statements

Finding who is the manager of which branch:

```
SELECT staff.staff_id, staff.fname, branch.branch_name
FROM staff
JOIN branch
ON staff.staff_id = branch.manager_id;
```

staff_id	fname	branch_name
2	Fahd	Sales
4	Angela	Repair
6	Bernie	Customer Service

Figure 10: Branches and their managers

Finding who has which manager:

```
SELECT staff.fname, branch.manager_id
FROM staff
JOIN branch
ON staff.supervisor_id = branch.manager_id
ORDER BY supervisor_id;
```

fname	manager_id
Fab	2
Patrick	2
Teddy	2
Sara	4
Coma	6

Figure 11: Some of the employees and their managers

Finding bike names and their types:

```
SELECT bike_info.bike_name, bike_type.type_name
FROM bike_info
JOIN bike_type
ON bike_info.category_id = bike_type.type_id;
```

bike_name	type_name
Surly Straggler 650b	Cyclocross Bicycles
Trek CrossRip+	Electric Bikes
Sun Bicycles Cruz 7	Cruisers Bicycles
Electra Amsterdam Original 3i	Cruisers Bicycles

Figure 12: Some of the bikes and their types

Finding who have sold more than 550 bucks (nested):

```
SELECT staff.staff_id, staff.fname, staff.lname
FROM staff
WHERE staff.staff_id IN (
    SELECT orders.staff_id
    FROM orders
    WHERE orders.order_id IN (
        SELECT ordered_items.order_id
        FROM ordered_items
        WHERE price > 550
    )
);
```

staff_id	First Name	Last Name
2	Fahd	Jaff
7	Abi	Gale
8	Patrick	Star
10	Sara	Michaels

Figure 13: Employees with sales more than 550 bucks

Finding Fahd's customers:

```
SELECT c.fname, c.city
FROM customers c
WHERE c.customer_id IN(
    SELECT o.customer_id
    FROM orders o
    WHERE o.order_id IN (
        SELECT order_id FROM orders
        WHERE staff_id = 2)
);
```

First Name	Last Name	city
Alperen	Duran	Krakow
Eden	Schreiner	Vechta

Figure 14: Customers who Fahd sold bikes and their cities

5 Conclusion

The user can easily find and update information about shops, bikes, customers. Moreover they can also follow the progress of the orders.

My thoughts:

Creating a database from scratch got quite complicated at some point so, I tried to plan a diagram by myself with the idea of the database first. Visualizing the concept made it easier to code. While trying to come up with select statements, it got easier to understand too. This project was a great opportunity for me to learn and understand the basics of SQL.