University of Mauritius

Mini Project Report
Database Design
Mrs **Sudha Cheerkoot-Jalim**

TOY SHOP DATABASE

Akshit Tooshalduth Sarju	2012925
Jeevesh Rishi Dindyal	2012397
Mounish Kumar Bholah	2013222

2021

Description of Database

✧ Overview

Daito Otaku Toy Shop, is a recently opened online shop that sells a wide variety of products at different price range, for all types of customers, from different suppliers. The aim of the shop is to help local people have ease of access to these products.

Instead of a paper-based database, which will be tedious to manage with the amount of information the Toy Shop opted for a computerized database. The database implemented stores different data for the shop that is the details about the products, customers, employees, shipment among many others.

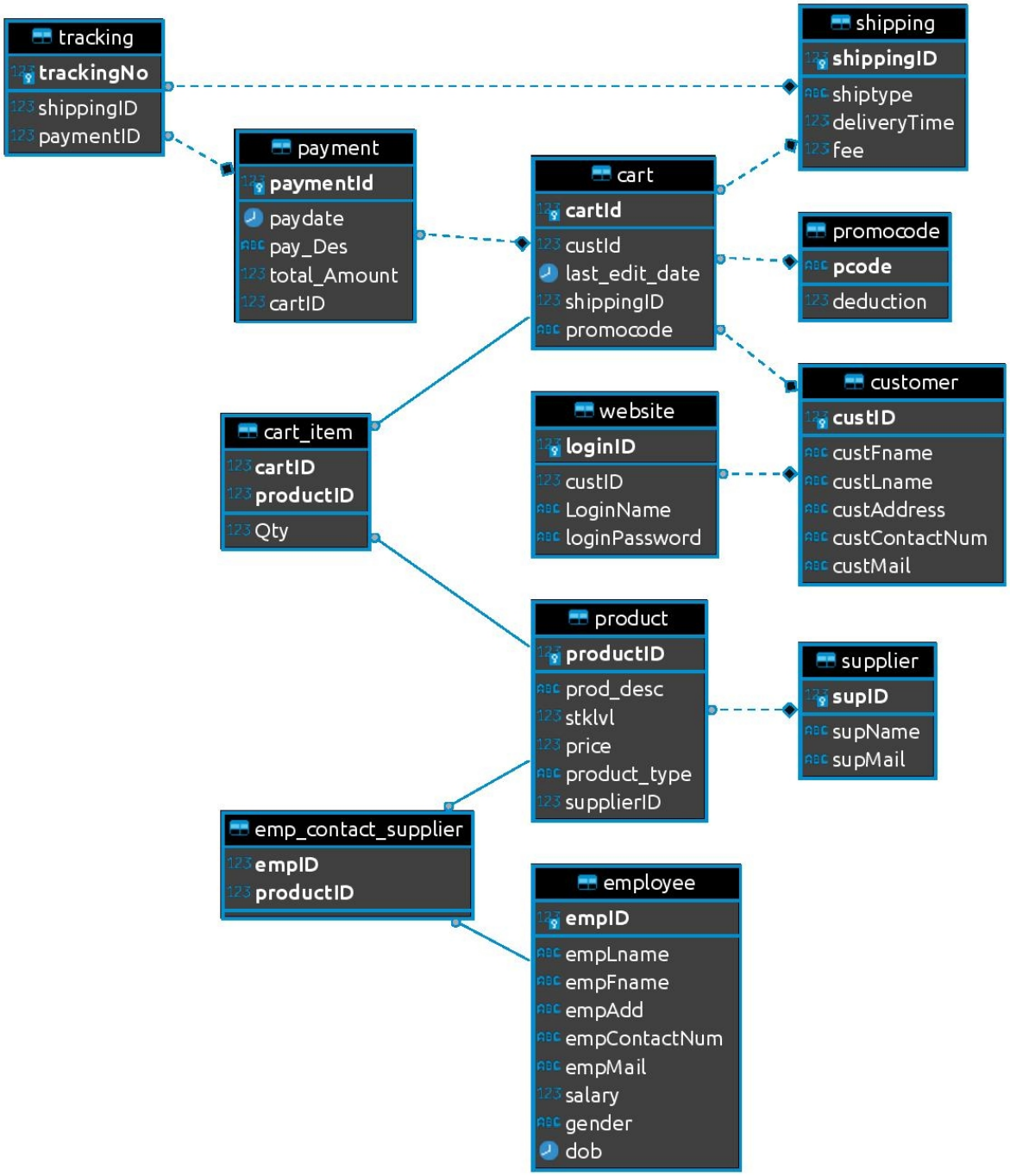
A computerized database is more appropriate since the database is quite complex as there are many attributes to be linked with one another to have a good database.

✧ How the database works?

Firstly, employees buy a product from a supplier. The data about employees and product are stored.
Each customer who wants to buy something from the website can be a visited or registered customer. For a registered customer login name and login password is used. Therefore data about the login password and name will be store.

Furthermore, when a customer enters the shop and chooses items to be bought, a cart is created and the items are stored and can continue shopping. When the customer decides to purchase the item(s) and proceed to payment a shopping and tracking number for the customer's order is assigned and there are reduction in price if a promo-code is applied. The amount of cost (cost of product + shipping - discount(if applicable)) and delivery time is given and hence, information about the cart, cart items, promotion, shipping and tracking are stored.

Entity Relationship Diagram [ERD]



Normalization

❖ *Functional Dependencies*

empID ---> empLname, empFname, empAdd, empContactNum, emp_Mail, salary, gender, dob
custId ---> custFname, custLname, custAddress, custContactNum, custMail
productID ---> prod_desc, stklvl, price, product_type, supplierID
supID ---> supName, supMail
trackingNo ---> shippingID, paymentID
cartID ---> custId, last_edit_date, shippingID, pcode, productID, Qty
paymentID ---> paydate, pay_Desc,total_Amount, cartID
promocode --> deduction
loginID ---> custID, LoginName, loginPassword
shippingID ---> shiptype, deliveryTime, fee

ONF

DaitoToyShop {(empID ,empLname, empFname, empAdd,
empContactNum, emp_Mail,salary,gender,dob,
custId , custFname, custLname, custAddress,
custContactNum, custMail,
productID, prod_desc, stklvl, price, product_type,
supplierID, supName, supMail,
trackingNo,
last_edit_date, Qty,
paymentID, paydate, pay_Desc,total_Amount,
pcode, deduction,
shippingID, shiptype, deliveryTime, fee,
loginID, custID, LoginName, loginPassword)}

1NF

DaitoToyShop {empID ,empLname, empFname, empAdd,
empContactNum, emp_Mail, salary, gender,
dob,
custId ,custFname,custLname,custAddress,
custContactNum, custMail,
productID, prod_desc, stklvl, price, product_type,
supplierID, supName, supMail,
trackingNo,
last_edit_date, shippingID, Qty,
paymentID, paydate, pay_Desc,total_Amount,
pcode, deduction,
shippingID, shiptype, deliveryTime, fee,
loginID, custID, LoginName, loginPassword}

- ***Primary Key: empID, loginID, trackingNo***

2NF

employeeDetails {empID ,empLname, empFname, empAdd,
empContactNum, emp_Mail, salary, gender, dob}

transactionDetails {trackingNo, shippingID, shiptype,
deliveryTime, fee, paymentID, paydate, pay_Desc,
total_Amount, cartID, productID, prod_desc, stklvl, price,
product_type, supplierID, supID, supName, supMail, Qty,
custId, custFname, custLname, custAddress, custContactNum,
custMail, last_edit_date, pcode, deduction}

websiteDetails {loginID, custID, custFname, custLname,
custAddress, custContactNum, custMail, LoginName,
loginPassword}

3NF

employee {(empID ,empLname, empFname, empAdd, empContactNum, emp_Mail, salary, gender, dob}

customer {custId, custFname, custLname, custAddress, custContactNum, custMail}

product {productID, prod_desc, stklvl, price, product_type, supplierID}

supplier {supID, supName, supMail}

cart {cartID, custId, last_edit_date, shippingID, pcode}

cart_item {cartID, productID, Qty}

payment {paymentID, paydate, pay_Desc, total_Amount, cartID}

promocode {pcode, deduction}

shipping {shippingID, shiptype, deliveryTime, fee}

tracking {trackingNo, shippingID, paymentID}

website {loginID, custID, LoginName, loginPassword}

emp_contact_supplier {empID, supID, productID}

Database Design

Tables / Fields / Data Types / Constraints

```
CREATE TABLE employee(
    empID INT IDENTITY PRIMARY KEY,
    empLname VARCHAR(20),
    empFname VARCHAR(20),
    empAdd VARCHAR(255),
    empContactNum VARCHAR(15),
    empMail VARCHAR(40) CHECK(empMail LIKE '%@%.%'),
    salary FLOAT CHECK(salary > 0),
    gender VARCHAR(1) CHECK(gender IN ('M','F','f','m')),
    dob DATETIME);

CREATE TABLE customer(
    custID INT IDENTITY(1,1) PRIMARY KEY,
    custFname VARCHAR(35) NOT NULL,
    custLname VARCHAR(35) NOT NULL,
    custAddress VARCHAR(255) NOT NULL,
    custContactNum VARCHAR(15),
    custMail VARCHAR(40) CHECK(custMail LIKE '%@%.%') NOT
NULL);

CREATE TABLE product(
    productID INT IDENTITY PRIMARY KEY,
    prod_desc VARCHAR(255),
    stklvl INT CHECK(stklvl > -1),
    price FLOAT,--Dollar
    product_type VARCHAR(50),
    supplierID INT NOT NULL,);

CREATE TABLE promocode(
    pcode VARCHAR(20) primary key,
    deduction FLOAT CHECK(deduction < 1));

CREATE table supplier(
    supID INT IDENTITY PRIMARY KEY,
    supName VARCHAR(35),
    supMail VARCHAR(30) CHECK(supMail LIKE '%@%.%') NOT
NULL UNIQUE,);
```

```
CREATE TABLE emp_buy_product(
    empID INT,
    productID INT,
    PRIMARY KEY(empID,productID));

CREATE TABLE website(
    loginID INT IDENTITY PRIMARY KEY,
    custID INT NOT NULL,
    LoginName VARCHAR(35) NOT NULL,
    loginPassword VARCHAR(20) NOT NULL);

CREATE TABLE cart(
    cartId INT IDENTITY PRIMARY KEY,
    custId INT NOT NULL,
    last_edit_date DATETIME NOT NULL,
    shippingID INT,
    promocode VARCHAR(20) NULL);


CREATE TABLE cart_item(
    cartID INT,
    productID INT,
    Qty INT check(Qty > 0),
    primary key(cartID, productID));

CREATE TABLE payment(
    paymentId INT IDENTITY PRIMARY KEY,
    paydate datetime NOT NULL,
    pay_Des VARCHAR(30),
    total_Amount FLOAT CHECK(total_Amount > 0) NOT NULL,
    cartID INT NOT NULL,);

CREATE TABLE shipping(
    shippingID INT IDENTITY PRIMARY KEY,
    shiptype VARCHAR(20),
    deliveryTime INT,
    fee FLOAT);

CREATE TABLE tracking(
    trackingNo INT IDENTITY PRIMARY KEY,
    shippingID INT NOT NULL,
    paymentID INT NOT NULL);
```

Relationships

```
ALTER TABLE cart
ADD FOREIGN KEY (custId) REFERENCES customer(custID)
ALTER TABLE cart
ADD FOREIGN KEY (shippingID) REFERENCES shipping(shippingID)
ALTER TABLE cart
add foreign key (promocode) REFERENCES promocode(pcode)

ALTER TABLE product
ADD FOREIGN KEY (supplierID) REFERENCES supplier(supID)

ALTER TABLE cart_item
ADD FOREIGN KEY (cartID) REFERENCES cart(cartID)
ALTER TABLE cart_item
ADD FOREIGN KEY (productID) REFERENCES product(productID)

ALTER TABLE website
ADD FOREIGN KEY (custID) REFERENCES customer(custID)

ALTER TABLE emp_buy_product
ADD FOREIGN KEY (empID) REFERENCES employee(empID)
ALTER TABLE Emp_contact_supplier
ADD FOREIGN KEY (productID) REFERENCES product(productID)

ALTER TABLE payment
ADD FOREIGN KEY (cartID) REFERENCES cart(cartID)

ALTER TABLE tracking
ADD FOREIGN KEY (shippingID) REFERENCES shipping(shippingID)
ALTER TABLE tracking
ADD FOREIGN KEY (paymentID) REFERENCES payment(paymentID)
```

Stored Procedures / INSERT

1.sp_customer_insert

- Insert for customers
- Customer first and last name, address, contact information is entered
- customer mail should be unique
- Displays appropriate message if successful or if an error occurred
- Login name and login password also inserted
- Displays appropriate message if successful or if null
- Login details can be null if the customer had a transaction in visited mode
- Else login details will be saved

2.sp_insert_supplier

- Insert for suppliers
- Supplier's name and email is registered
- Displays appropriate message if successful or if an error occurred

3.sp_insert_promocode

- Insert for promocodes
- Promocode and percentage deduction is input
- Displays appropriate message if successful or if an error occurred

4.sp_insert_employee

- Insert for employees information
- Employees first and last name, address, contact information, gender, salary and date of birth is stored
- Displays appropriate message if successful or if an error occurred

5.sp_insert_shipping

- Insert for shipping information
- Shipment type, delivery time and shipping cost is input
- Displays appropriate message if successful or if an error occurred

6.sp_insert_product

- Insert for product details
- Product description, unit price, type and stock level is input
- Displays appropriate message if successful or if an error occurred
- Supplier ID is also inserted
- Displays appropriate message if successful or if supplier ID not found

7.sp_insert_user

- Insert for login details for website
- Displays appropriate message if successful or if an error occurred
- It checks if the person is a customer as well and if not an error is displayed

8.sp_insert_cart

- Insert for cart details
- Customer ID and Cart ID is inserted
- Displays appropriate message if successful or if the customer ID does not exist or if quantity requested by the customer is too high compared to stock level

9.sp_insert_payment

- Insert for payment transaction information
- Payment description, cart ID is entered
- It checks if the cart has already been paid
- It calculates the total cost for the transaction by adding the product cost, shipping fee and minus the deduction is any.
- Displays appropriate message if an error occurred

10.sp_insert_emp_product

- Insert for employees, product
- This shows which employee contacted which suppliers
- And which products come from which supplier
- Displays appropriate message if successful or if employee ID or supplier ID does not exist

11.sp_insert_tracking

- Insert for tracking details
- Payment ID, shipping ID are inserted
- Displays appropriate message if successful or if payment ID or shipping ID does not exist

12.sp_insert_cartItem

- Insert for cart items details
- Input what are there in specific carts
- Cart ID, Product ID and quantity of products are entered
- Displays appropriate message if successful or if an error occurred
- It also checks if a customer add more than there is in stocks in cart. A message is prompted

Stored Procedures / Mini Queries

1.sp_productQty_totalprice
- Display the total price of products for the whole stock

2.sp_productQty_totalprice_specific_item
- Display the total price of a specific product for the whole stock

3.sp_chart_total_cost
- Display the total cost in a cart of a customer

4.sp_check_paid
- Display all that have paid for their products in the cart

5.sp_price_range
- Display the price ranges when specified the range in ascending order [From lowest to highest]

6.sp_check_active_cart
- Displays all the carts that are active that is payment has not been made

7.sp_stock_price
- Displays the unit price of products

8.sp_in_shop_deliver
- Displays all the transaction that will be taken at the shop [No shipping required]

9.sp_check_need_to_ship
- Displays all the payment ID which needs to be shipped
- If an item is in tracking, it means it has already been delivered or already shipped
- But if it is not in tracking then it has not shipped yet

10.sp_highest_sale
- Displays the highest sale the shop has made

11.sp_stock_price_specific
- Displays the unit price of a product when specified

12.sp_del_cust

- When executed it delete a specified customer records as well as its login details

13.sp_checks_if_exist

- Displays if the input ID of employee, customers, supplier, products input exists or not

Triggers

1.tg_check_customer
- Checks if customer is already added in database

2.tg_tg_check_email_customer
- Checks if mail is valid
- It checks if email address has @ symbol or it displays an error message
- check if email address already in use

3.tg_check_product
- Checks if product is already added in database
- if added, product stock level is update with new stock level being addition of both stock level of insert and product

4.tg_increase_count_products
- Increment number of products when added in a table

5.tg_check_supplier
- Checks if supplier is already added in database

6.tg_disp_prod_details
- Displays a message with the description of the new product added

7.tg_store_old_data
- Stores old data about the price when it changes in an audit table

8.tg_old_products
- Stores data about products that are not sold anymore in the shop in an audit table

9.tg_updte_stk
- Reduces the stock level when the payment of a cart is done by the quantity bought by the customer

10.tg_count_customer
- Increment number of customers when added in a table

11.tg_check_cart

- Displays an message if a customer is has already a cart when another cart is being assign to the person

12.tg_check_cart_payment

- Displays a message if items are placed in a cart which has already been paid.
- When a customer add a product to paid cart then;
    - ✓ new cart is created
    - ✓ Customer ID, shipping ID and promocode is inserted in the cart table
    - ✓ Cart ID, product ID and quantity is inserted into cart_item table with new cart ID in insert

13.tg_check_cart

- When an item gets inserted in a cart the last edit time gets updated.

Other Design Considerations

i. Identity Primary Key is used to auto generate primary keys of tables like product ID, customer ID.

ii. Data can be inserted from the stored procedure created for each table to check various constraints without the need to create triggers.
For instance, checking the email address if it contains @ symbol or if a product or supplier or customer is already in the database.

iii. One customer can have multiple accounts to login.

iv. Even though the shop is online and has shipping people can come to collect their purchased item(s) in the shop store.

v. Carts, even though they contain products, gets deleted after 2 weeks of inactivity.

vi. Promo-codes cannot be used twice for the same cart.

vii. Stock of products are reduced only after payment not when added to cart.

viii. Promo-codes are like coupons and the discount is applied when buying the products.
The sellers can decide if they want to give them a discount or not [Depends on purchase size, for example discount may be given if price is over $15].

ix. Not all items are shipped after purchase but have to be assigned a tracking number and then shipment begins.

Work Division