3 MAY 2023

CS3120 - DATA BASE MANAGEMENT SYSTEM LAB

ONLINE SHOPPING CART DATABASE

FINAL PROJECT REPORT

Collaborators:

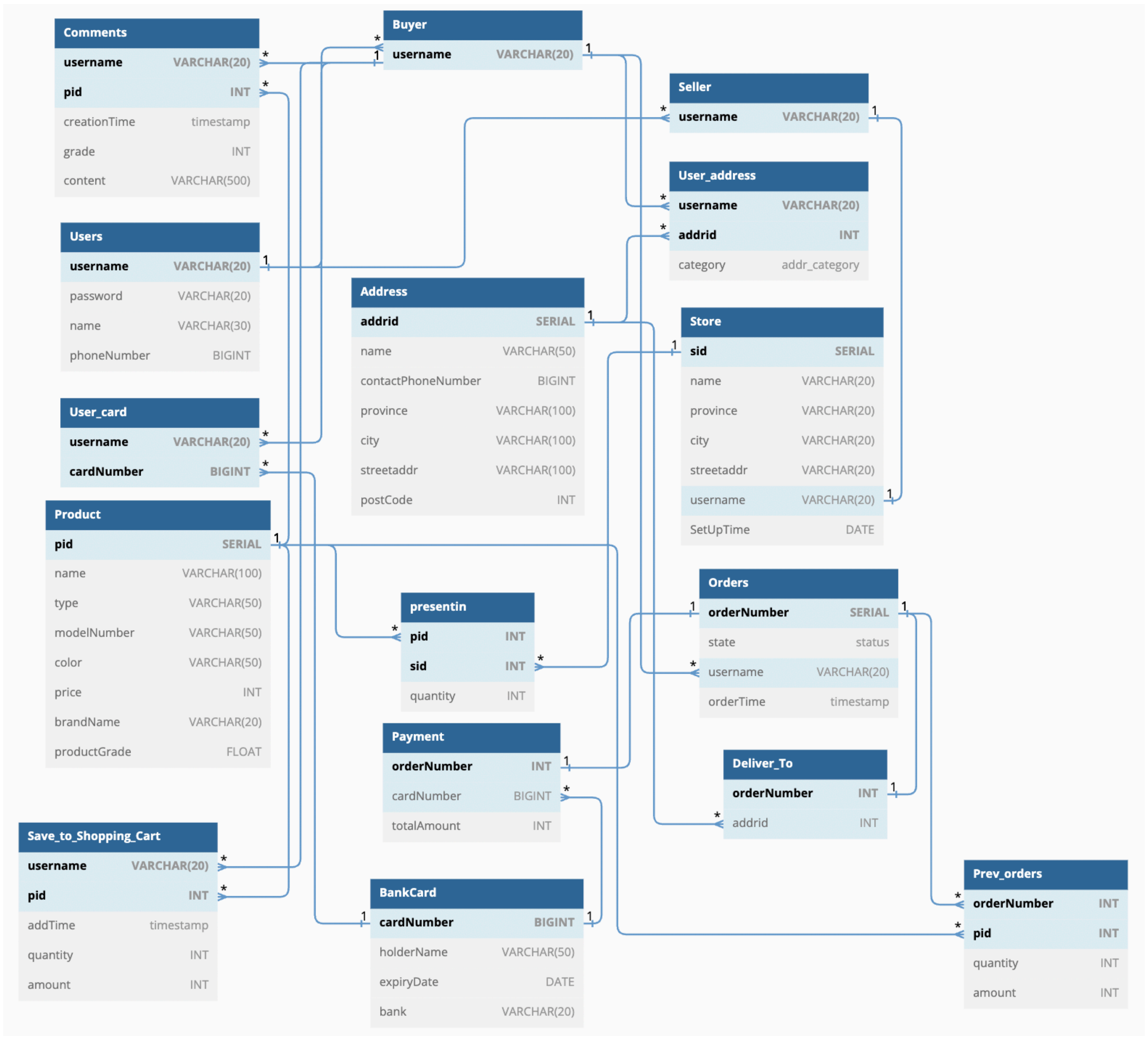
LIKHITH	112001021
SATHVIK	112001005
PAVAN	112001025

INTRODUCTION

The advent of the internet has revolutionised the way we shop. With the growth of e-commerce, online shopping has become more convenient and accessible than ever before. Online shopping websites enable customers to browse through a wide range of products from the comfort of their homes and purchase them with just a few clicks.

The online shopping cart database system is a vital component of any e-commerce website, as it enables customers to add products to their virtual carts and proceed to checkout. We present the design, implementation, and testing of an online shopping cart database system as a DBMS project. The project aims to create a functional and efficient database system that can be used to manage the online stores.

RELATIONAL MODEL



In the above relational model:

1. All the relations with attributes and their respective datatypes along with keys and connections are mentioned.

2. Primary key are represented by bold attributes in respective relations.

3. It is also mentioned that wether connections [foreign key references] are many-one or one-one.

INTEGRITY CONSTRAINTS

USERS

This table represents user details. Username has to be unique for all the users hence username is the primary key. All the other attributes have NOT NULL constraint except for phoneNumber attribute. There are CHECK constraints on username, password and phoneNumber attributes to make sure that username has at least 4 characters, password has at least 8 characters and phoneNumber has 10 digits.

BUYER

This table represents wether the user is a buyer. It has only single attribute which is username and is primary key and also foreign key which is referenced from Users table

SELLER

This table represents wether the user is a seller. It has only single attribute which is username and is primary key and also foreign key which is referenced from Users table

BANKCARD

This table represents details of bank cards saved by any buyer in the database. Since card number is unique to each card it is primary key and also has a CHECK constraint to check that it has 12 digits. All other attributes have NOT NULL constraints.

USER_CARD

Same bank card can be saved by many users and a user can save many cards and this table captures that relation. Username and cardNumber combined is a primary key. Username is a foreign key referenced from Buyer table and cardNumber is a foreign key referenced from BankCard table.

ADDRESS

This table represents details of addresses saved by any buyer in the database. Since each address have unique id, addrid is the primary key. Attributes of name, streetaddr and postCode have NOT NULL constraints. There is CHECK constraint on postCode and contactPhoneNumber to check that postcode is 6 digits and phone no is 10 digits.

USER_ADDRESS

Same address can be saved by many users and a user can save upto 4 addresses and is captured by this table. Username and addrid combined is the primary key. Attribute category is of enumerator datatype addr_category [can take values of home, work, friend or other] and has NOT NULL constraint to it. Username is a foreign key referenced from Buyer table and addrid is foreign key referenced from Address table.

STORE

This table captures store details of a seller. Since each store have unique id, sid is the primary key. Attributes name, username and setupTime have NOT NULL constraints. Username has UNIQUE constraint. SetupTime has default value of CURRENT_DATE. Username is a foreign key referenced from Seller table.

PRODUCT

This table captures details of products in the database. Since each product have unique id, pid is the primary key. Attributes name, modelNumber and price have NOT NULL constraints. ModelNumber also has UNIQUE constraint. There is a CHECK constraint on productGrade so that it is less than or equal to 5.

PRESENTIN

This table represents quantity of any product available at different stores. Sid and pid combined is the primary key. Quantity has NOT NULL constraint. Sid is a foreign key referenced from Store table and pid is a foreign key referenced from Product table.

ORDERS

This table represents orders made by any user. Since orderNumber is unique to each order, it is the primary key. All attributes have NOT NULL constraint. Default value of orderTime is CURRENT_TIMESTAMP. State is of enumerator datatype status [can take values of success or failed]. Username is a foreign key referenced from Buyer table.

COMMENTS

This table represents grading/commenting on any product by any user. Username and pid combined is the primary key. Grade and creationTime have NOT NULL constraint. CreationTime has default value of CURRENT_TIMESTAMP. Grade has a CHECK constraint on it to check that it is less than or equal to 5. Username is a foreign key referenced from Buyer table and pid is a foreign key referenced from Product table.

SAVE_TO_SHOPPING_CART

This table represents the shopping cart of any user. Since each user can have multiple products saved in their cart, Username and pid combined is the primary key. All attributes have NOT NULL constraint. AddTime has a default value of CURRENT_TIMESTAMP. Username is a foreign key referenced from Buyer table and pid is a foreign key referenced from Product table.

PAYMENT

This table represents through which bank card the order has been placed. Since for each order placed payment can only be done once, orderNumber is the primary key. TotalAmount has NOT NULL constraint. OrderNumber is a foreign key referenced from Orders table and cardNumber is a foreign key referenced from BankCard table.

DELIVER_TO

This table represents to which addresses the order has been delivered. Since each order is delivered only once, orderNumber is the primary key. OrderNumber is a foreign key referenced from Orders table and addrid is a foreign key referenced from Address table.

PREV_ORDERS

This table represents details of orders made till now [success or failed] by any user. Since each order may contain various products, orderNumber and pid combined is the primary key. All attributes have NOT NULL constraint. OrderNumber is a foreign key referenced from Orders table and pid is a foreign key referenced from Product table.

VIEWS

USER_DETAILS

This view is created to extract the user details namely name and phoneNumber of the current user from Users table.

USER_ADDRESSES

This view is created to extract name, contactPhoneNumber, province, city, streetaddr, postCode, category From User_address NATURAL JOIN Address for the current user.

USER_BANKCARD

This view is created to display the card details namely holderName, cardNumber, expiryDate, bank from User_card NATURAL JOIN BankCard for the current user.

COMMENTS_PRODUCT

This view is created to see the pid, grade, content from Comments NATURAL JOIN Users for the current user [buyer]. This is for buyer to add/edit their comments.

BUYER_SAVE_TO_CART

This view is created to view the pid, quantity from Save_to_Shopping_Cart for the current user [buyer]. This is for buyer to view/edit their shopping cart.

BUYER_ORDER_DETAILS

This view is created to select orderNumber, totalAmount, state, orderTime from Orders NATURAL JOIN Payment for the current user [buyer]. This is for buyers to view list of their orders made so far [success or failed].

PRODUCT_SELLER

This view is created to extract sid, pid, name, type, modelNumber, color, brandName, price, quantity FROM product NATURAL JOIN presentin NATURAL JOIN Store for the current user [seller]. This is for seller to view/add/edit their products they are selling.

STORE_SELLER

This view is created to get the name, province, city, streetaddr, SetUpTime from Store for the current user [seller]. This is for seller to view/edit their store.

FUNCTIONS AND TRIGGERS

CATEGORY_SELECTION

This function is used to return all the distinct categories of products available in the database. It has no input parameters and returns a single column table of distinct categories.

BRAND_SELECTION

This function is used to return all the distinct brand names of products for a category selected in the database. It has one input parameter of datatype VARCHAR(50). It outputs a single column table of all brands available for the category selected. If category selected is NONE then, it outputs all the brands of all products from all categories.

SAVING_TO_CART

This is a trigger to update Save_to_Shopping_Cart table when buyer inserts new products onto their cart. It executes function update_cart instead of insert on Buyer_Save_to_Cart view. This triggers when buyer browses any product and add that to their cart.

UPDATE_CART

This is a trigger function checks wether the product already exists in cart or not. If product already exists then it increases the quantity in cart by amount selected. If the product does not exists then it adds this new product to cart with quantity selected.

UPDATING_TO_CART

This is a trigger to update Save_to_Shopping_Cart table when buyer changes the quantity of products in his cart. It executes function updating_cart instead of update on Buyer_Save_to_Cart view. This triggers when buyer changes quantities of products from their cart page.

UPDATING_CART

This is a trigger function that updates quantity of products in cart by new quantity. If new quantity is zero then, it deletes that product from the cart.

TRANSACTIONS2

This is a function that takes no parameter and returns integer. When user places his order from cart, this function checks wether quantities of all products ordered does not exceed available quantity. If exceeds then it returns 1 else it returns 0.

TRANSACTIONS

This is function that takes two parameters, address category of enum type addr_category and card number of type bigint and returns integer. When buyer proceeds to select address and bankcard to place order, this function checks selected parameter and if any of two inputs are missing then it considers order status as failed and inserts into Orders table and prev_orders table correspondingly with status as failed. If both fields are selected then it places order and inserts into Orders, Prev_orders, Payment and Deliver_to tables correspondingly with status success and payment, delivery address details and also refreshes the buyer's cart.

USER_UPDATE

This is a trigger to update Users table when a user updates their personal information [name or phone number]. It executes function user_det_update instead of update on user_details view.

USER_DET_UPDATE

This is trigger function that updates name/phone number attributes of Users table of corresponding user to new name/new phone number respectively.

USER_ADDR_ADD

This is a trigger to update/insert into Address and User_address tables when buyer adds new address to their account. It executes function update_addr instead of insert on User_addresses view.

UPDATE_ADDR

This is trigger function that checks wether given address already exists in database or not. If not exists then it inserts into Address table and gets new address, else it gets address id from Address table. Now it checks wether selected address category by buyer already has an address listed under it. If not then it inserts address id, category into User_address table under buyer username, else it updates address id in User_address table record for selected category under this user to new address id.

USER_CARD_ADD

This is a trigger to update/insert into BankCard and User_card tables when buyer adds new bank card to their account. It executes function update_card instead of insert on User_bankcard view.

UPDATE_CARD

This is trigger function that checks wether given address already exists in database or not. If not exists then it inserts into Address table and gets new address, else it gets address id from Address table. Now it checks wether selected address category by buyer already has an address listed under it. If not then it inserts address id, category into User_address table under buyer username, else it updates address id in User_address table record for selected category under this user to new address id.

COMMENT_ADD

This is trigger to update/insert into Comments table when buyer comments on any product. It executes function update_comment instead of insert on Comments_product view.

UPDATE_COMMENT

This a trigger function that check wether there exists comment already made by user on the product. If already exists then it updates Comments table, else it inserts into Comments table with new grade or/and new comment made by user on a product. Since new grade has been given to product, It then calculates new average grade for the product and updates grade attribute of product in Product table.

STORE_UPDATE

This is trigger to update/inert into Store table when seller adds their store details for the first time or edits their store details. It executes function seller_new_prod instead of insert on store_seller view.

SELLER_UPDATE_STORE

This is trigger function that checks wether there is a store associated with seller or not. If seller has no store listed under them then it inserts store details and username of seller into Store table, else it updates the store details of the seller from Store table to new details provided.

PRODUCT_INSERT

This is trigger to inert into Product and presenting table when seller adds new products to their store. It executes function seller_new_prod instead of insert on product_seller view.

SELLER_NEW_PROD

This is trigger function that checks wether the product already exists in database. If exists then it gets pid of product, else it inserts new product into product table and gets new pid. Now it checks wether this product already exists in the store. If exists then it raises exception saying product already exists, else it inserts pid, sid of store and quantity into presentin table. Before inserting it also checks wether there is a store associated with seller or not. If seller has no store listed under them then it raises exception saying enter store details before adding any product.

INDICES AND FREQUENT QUERIES

List of Indices

INDEX name	Type Of Index	On Table	On Attribute
user_name_idx	HASH Index	Users	username
buyer_idx	HASH Index	Buyer	username
user_address_idx	HASH Index	User_address	addrid
Address_idx	HASH Index	Address	addrid
usercard_idx	HASH Index	User_card	username
user_card_idx	HASH Index	User_card	cardNumber
bankcard_idx	HASH Index	BankCard	cardNumber
comments_idx	HASH Index	Comments	username
save_to_shopping_idx	HASH Index	Save_to_Shopping_Cart	username
orders_idx	HASH Index	Orders	orderNumber
payment_idx	HASH Index	Payment	orderNumber
seller_idx	HASH Index	Seller	username
deliver_to_idx	HASH Index	Deliver_To	addrid
store_idx	HASH Index	Store	username
product_idx	HASH Index	Product	pid
prev_orders_idx	HASH Index	Prev_orders	pid

1.
   ```
   SELECT name, phoneNumber
   FROM Users
   WHERE username = CURRENT_USER;
   ```

This query will use the user_name_idx with equality on username.This query is used to retrieve the name and phoneNumber of the current user.We used this query in User_details view.

2.
   ```
   SELECT name, contactPhoneNumber, province, city, streetaddr, postCode,
   category
   FROM User_address NATURAL JOIN Address
   WHERE username = CURRENT_USER;
   ```

This query will use user_address_idx and Address_idx indices to retrieve the records of the current user with equality on attribute username.we used this query in User_addresses view.

3.
   ```
   SELECT holderName, cardNumber, expiryDate, bank
   FROM User_card NATURAL JOIN BankCard
   WHERE username = CURRENT_USER;
   ```

This query will use user_card_idx and bankcard_idx indices to retrieve the records with equality on attribute cardNumber.We used this query in User_bankCard view.

4.
   ```
   SELECT pid, grade, content
   FROM Comments NATURAL JOIN Users
   WHERE username = CURRENT_USER;
   ```

This query will use the comments_idx and user_name_idx indices to retrieve the records with equality on the attribute username.We used this query in Comments_product view.

ROLES AND PRIVILEGES

MANAGER

Manager has permissions to all the operations on all the tables in the schema

BUYER

Buyer has SELECT ON Product, Buyer_Order_details, Comments,Prev_orders and SELECT, INSERT, UPDATE, DELETE ON Comments_product, User_addresses, Buyer_Save_to_Cart, User_details, User_bankCard.

SELLER

Seller has SELECT ON Comments table and
SELECT, INSERT, UPDATE, DELETE ON User_details, Product_seller, store_seller.

REGISTERED ROLES

We as a user can create an account as either seller or buyer. If we create the account as buyer then a role is created under that username and it falls under the BUYER group, that is GRANT BUYER to the role created. If we create the account as seller then a role is created under that username and it falls under the SELLER group, that is GRANT SELLER to the role created.

GENERAL QUERIES

1. Select count(*) from category_selection();

To get total no.of category of products available in the database.The buyer can run this query.

2. Select count(*) from brand_selection(‘Audio’);

To get total no.of brands that are available in the database in the given category(‘Audio’). The buyer can run this query. Here we provided input of Audio and we get number of brands available in the category. If we don’t provide any input to function then this query will count no.of brands in all the categories.

3. Insert into Buyer_Save_to_Cart values(23,4);

This query will call a trigger saving_to_cart and that trigger will run a function update_cart(). This will update/add product with pid 23 to respective buyer’s cart with quantity of 4. This can be run by manager.

4. Insert into store_seller values(‘subhash’,’deccan’,’hyderabad’,’charminar’);

This query will call a trigger store_update and that trigger will run a function seller_update_store() .This will update store details of the seller with given details.

5. Insert into product_seller(‘vita’,’Food’,’1234’,’brown’,40,’Catberry’,200);

This query will call a trigger product_insert which in turn run function seller_new_prod().This function will insert new product into the seller store. In above query this adds new product vita of category Food with quantity of 200 to store.