DATABASE DESIGN FINAL PROJECT REPORT

DATABASE DESIGN FOR

EBAY

TABLE OF CONTENTS

INTRODUCTION

The aim of the project is to create a database for E-Commerce company E-Bay. This covers operations like transactions between buyer and seller and all other basic functionalities of the Database. The data requirements are as below:

DATA REQUIREMENTS

1. User: A new user can create an account and a user can be a seller or a buyer.
2. Product: A product can be added by a seller. He has two options to sell, he may directly sell it or auction the product. Product will have description, color, pictures
3. Category: Products are classified into various categories like electronics, clothing etc.
4. Offers: Products can have offer. A seller may give an offer on direct buy or auction as well
5. Cart: A Buyer can directly add the product to his cart
6. Wishlist: A buyer can also add the product to the Wishlist
7. Order Details: A Buyer can place an order and checkout by making the payment
8. Shipment: Products may be shipped through different carriers.
9. History: A buyer history is stored, where all the previous orders are recorded
10. Review: A buyer can provide the review for the product which may include comments and he has option to provide review for delivery and item as well.
11. Return: Buyer can also return the item by providing comments
12. Carrier: Order is shipped by a carrier
13. Bid: A seller can bid a product or may directly sell it
14. Direct Sell: A seller can bid a product or can sell it directly
15. Payment Info: Each checkout has a respective payment information saved for the transaction.

CARDINALITIES

1. **Buyer-Cart:** cardinality is 1:1 A buyer can have a single cart.
2. **Buyer-Seller:** cardinality is M: N, A Buyer can sell M products and product can be bought by N buyers
3. **Seller-Product:** cardinality is 1: N, A seller can sell N products and a product can be sold by only one seller
4. **Buyer-Wishlist:** cardinality is 1:1, A buyer can have one Wishlist and a Wishlist belongs to one buyer.
5. **Buyer-review**: cardinality is 1: N, a Buyer can give N reviews and a review is given by one Buyer.
6. **Cart-Product:** cardinality is M: N, a cart can have N products and a product can be in M carts
7. **Buyer-History:** cardinality is 1:1, a Buyer can have a single history and a history can have a single buyer
8. **Order Details-Returns:** cardinality is 1: N, an order can have N returns but a return will only have one respective order.
9. **Checkout-Order Details:** cardinality is 1:1, a checkout can have one order details and one order details can have one checkout
10. **Wishlist-Product:** cardinality is M: N, a Wishlist can have N products and a product may be present in M wish lists
11. **Buyer-Payment Info:** cardinality is 1:1, a Buyer can have 1 payment info and a payment info can have one respective Buyer.

Similarly other cardinalities were defined on system.

ER/EER-Diagram Access Link:

Draw.io: https://drive.google.com/file/d/16mPTgAUajQdxbg-Eo7ipSF2js6T7Y1U1/view?usp=sharing

Modeling of Requirements as ER/EER-Diagram

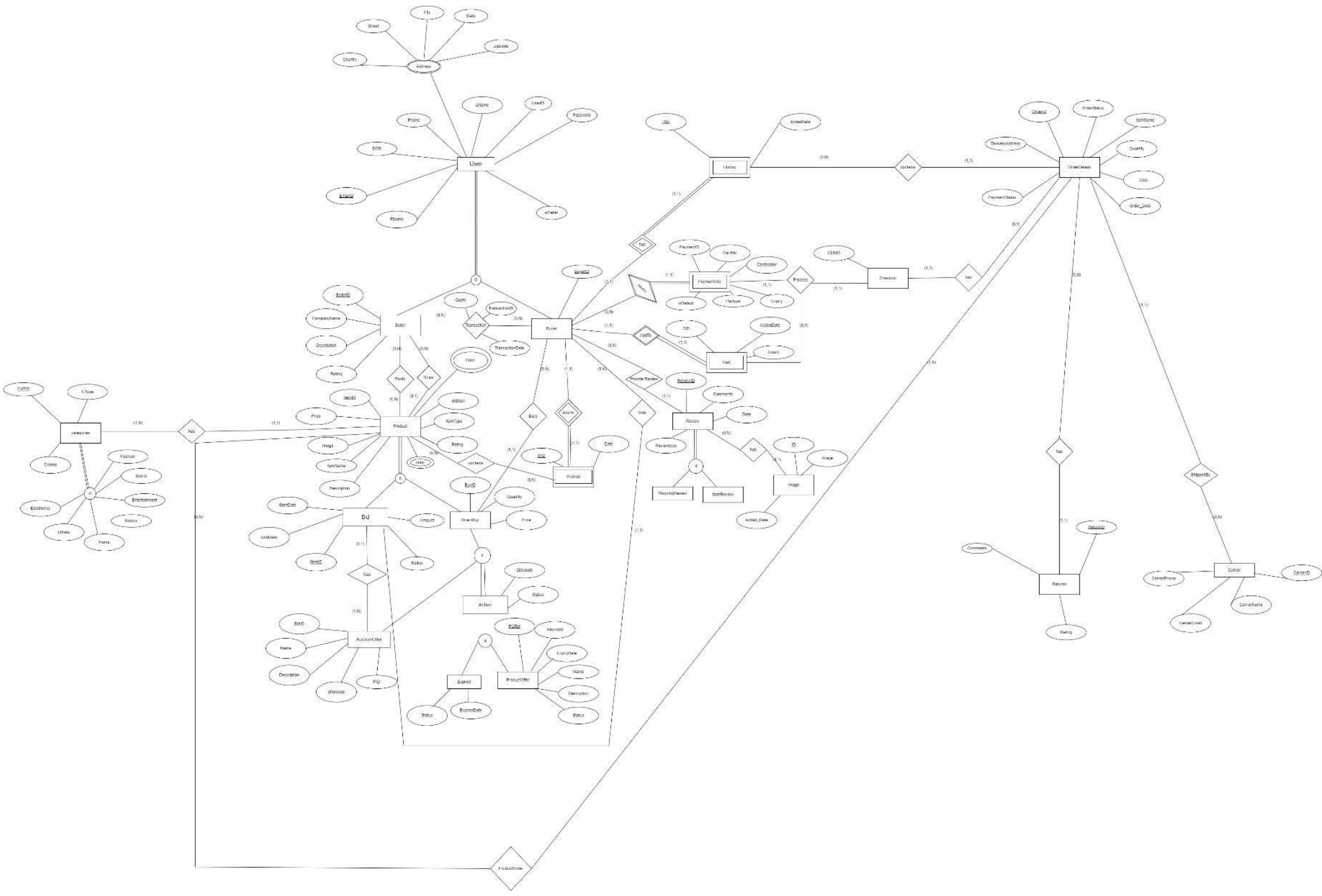


Fig 1.0 eBay’s Entity Relation Diagram (ER-Diagram)

Tables and Keys used:

Table Name	Primary Key	Foreign Key
Buyer	Buyer ID	BuyerID -> EmalID(users)
Seller	Seller ID	SellerID -> EmalID(users)
users	Email	
Address	MailId,Street,City,State,Country,ZipCode	MailID->EmailID(users)
Transaction	Buyer ID, Seller ID	BuyerID -> EmalID(users) SellerID -> EmalID(users)
Product	ItemID	SellerID -> EmalID(users) CATID->CATID(Category)
Category	CATID	
BID	ItemID	ItemID->ItemID(Product)
Product Offer	Poffer	
Epired	PID	PID->Poffer(ProductOffer)
Active	PID	PID->Poffer(ProductOffer
BuyerBIDS	BID	BID->ItemID(BID)
PaymentInfo	BuyerID,PaymentID	BuyerID -> EmalID(users)

OrderDetails	OrderID	UserID -> EmalID(users)
		HistoryID->HID(History)
		CID->CID(Cart)

History	HID	UserID -> EmalID(users)
Returns	ReturnID	OrderID -> OrderID(OrderDetails)
Carrier	CarrierID	
Cart	CID	
Checkout	CHKID	CID->CID(Cart)

Review	ReviewID	
Wishlist	WID	BuyerID -> EmalID(users)

WishlistContains	WID,ItemID	WID->WID(Wishlist)
		ItemID->ItemID(Product)

RELATIONAL SCHEMA

We will map the ER diagram into relational Schema

The Relation and structure are defined by relation schema. It consists of relation name, attributes, column names, filed names.

To convert ER to relational Schema we have rules:

Mapping of regular entities: Include all simple attributes in R and make one key attribute as primary key

Mapping of weak entities: Include all simple attributes in R and include the primary key of owner of the weak entity make the combination as primary key

Mapping 1:1 relation: choose an entity type with total participation include a foreign key in that relation the primary key of other relation.

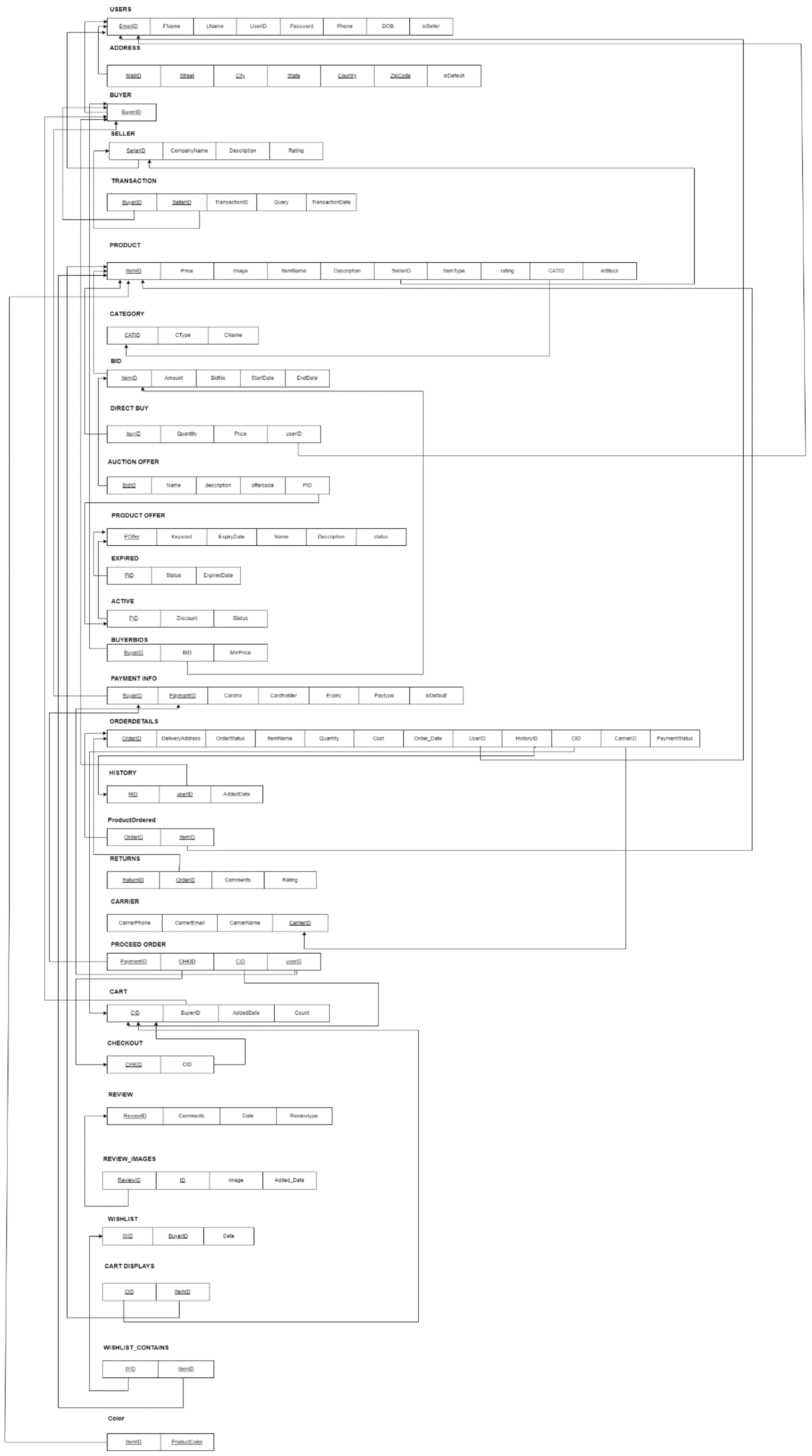
Mapping 1: N relation: Identify a relation that is on N-side of the relationship type include a foreign key in that relation the primary key of other relation

Mapping M: N relation: create a new relation with foreign key as the primary key of two participating entities and their combination will form a new primary key.

Mapping multivalued attribute: Create a new relation with the attribute

Relational Schema Access Links:

Draw.io: https://drive.google.com/file/d/1A36nZ-ibiq0Tw0SAx5dPMYWFF8LtXp8R/view?usp=sharing



Fix 2.0 Mapping ER Diagram into Relational Schema

Normalization of Relational Schema

The process of decomposing unsatisfactory relations by breaking up their attributes into smaller relations is called Normalization.

1NF: The attributes should hold atomic values. Multivalued, Composite attributes and their combinations are not allowed.

2NF: Every non-prime attribute A in R is fully functionally dependent on primary key

3NF: A relation should be 2NF and no non-prime attribute B in R is transitively dependent on primary key.

The Relational Schema that is being generated from our ER follows all three normalization rules and thus it is in 3NF already.

SQL

Statements to create Relations in DB and Add Constraints

```
CREATE TABLE USERS (
        EmailID VARCHAR (30),
        FName VARCHAR(30),
        LName VARCHAR(30),
        UserID VARCHAR(20),
        Password VARCHAR(20),
        Phone INT,
        DOB Date,
        isSeller BOOLEAN DEFAULT 0,
        PRIMARY key(EmailID)
)
CREATE TABLE ADDRESS (
        MailID VARCHAR (30),
        Street VARCHAR(20),
        City VARCHAR(20),
        State VARCHAR(20),
        Country VARCHAR(20),
        Zipcode INT,
        isDefault Boolean DEFAULT NULL,
        PRIMARY key(MailID, Street1, Street2, City, State, Country, Zipcode),
        FOREIGN key(MailID) REFERENCES USERS (EmailID) ON DELETE CASCADE
)
CREATE TABLE BUYER (
        BuyerID VARCHAR (30),
        PRIMARY key (BuyerID),
        FOREIGN key(BuyerID) REFERENCES USERS (EmailID) ON DELETE CASCADE
)
CREATE TABLE SELLER (
        SellerID VARCHAR (30),
        CompanyName VARCHAR (30),
        Description VARCHAR (30),
        Rating INT,
        PRIMARY key (SellerID),
        FOREIGN key(SellerID) REFERENCES USERS(EmailID) ON DELETE CASCADE
)

CREATE TABLE TRANSACTION (
        BuyerID VARCHAR (30),
        SellerID VARCHAR (30),
        TransactionID VARCHAR (30),
        Query VARCHAR (30),
        TranactionDate Date,
        PRIMARY key (BuyerID ,SellerID),
        FOREIGN key(BuyerID) REFERENCES buyer (BuyerID) ON DELETE CASCADE,
        FOREIGN key(SellerID) REFERENCES seller (SellerID) ON DELETE CASCADE
)
```

```
CREATE TABLE CATEGORY (
        CType INT,
        CName VARCHAR (20),
        CATID INT NOT NULL,
        PRIMARY key (CATID)
)
CREATE TABLE PRODUCT (
        ItemID INT NOT NULL,
        Price INT,
        Image VARCHAR (20),
        ItemName VARCHAR (30),
        Description VARCHAR (50),
        SellerID VARCHAR (30),
        ItemType VARCHAR (30),
        Rating INT,
        CATID INT,
        inStock CHAR(1),

        PRIMARY key (ItemID),
        FOREIGN key(CATID) REFERENCES CATEGORY(CATID) ON DELETE
        CASCADE,
        FOREIGN key(SellerID) REFERENCES SELLER (SellerID) ON DELETE CASCADE
)
CREATE TABLE BID (
        ItemID INT,
        Amount INT,
        BidNo INT,
        StartDate DATE,
        EndDate DATE,
        PRIMARY key (ItemID),
        FOREIGN key(ItemID) REFERENCES product (ItemID) ON DELETE CASCADE
```

```
)
CREATE TABLE DIRECT_BUY (
        BuyID INT,
        Quantity INT,
        Price INT,
        BuyerID VARCHAR (30),
        PRIMARY key(BuyID),
        FOREIGN key(BuyID) REFERENCES Product (ItemID) ON DELETE CASCADE
)


CREATE TABLE PRODUCT_OFFER (
        POffer INT,
        Keyword VARCHAR(15),
        ExpiryDate DATE,
        Name VARCHAR (20),
        Description VARCHAR (30),
        Status BOOLEAN DEFAULT 0,
        PRIMARY key(POffer)
)

CREATE TABLE ACTIVE(
        PID INT,
        DISCOUNT INT,
        STATUS CHAR(30),
        PRIMARY KEY(PID),
        FOREIGN key(PID) REFERENCES PRODUCT_OFFER(POffer) on delete CASCADE
)

CREATE TABLE AUCTION_OFFER (
        PID INT,
        BidID INT,
        OfferCode varchar(10),
        Description varchar(20),
        PRIMARY key(pid,bidID),
        FOREIGN key(pid) REFERENCES ACTIVE(pid) on DELETE CASCADE,
        FOREIGN key(bidID) REFERENCES BID(itemID) on DELETE CASCADE
)
CREATE TABLE EXPIRED(
        PID int(10),
        Status boolean,
        ExpiredDate Date,
        PRIMARY KEY(PID),
        FOREIGN KEY(PID) REFERENCES PRODUCT_OFFER(POffer) on delete CASCADE
)

CREATE TABLE BUYERBIDS(
        BID INT,
        buyerid varchar(50),
        BidDate date,
        PRIMARY key(bid, buyerid, bbid),
        FOREIGN key(bid) REFERENCES bid(itemid) on DELETE CASCADE,
        FOREIGN key(buyerid) REFERENCES buyer(buyerid) on DELETE CASCADE
)


CREATE TABLE PAYMENTINFO(
        buyerID varchar(50),
        PaymentId INT,
        cardno INT,
        Emonth INT,
        Eyear INT,
        PType varchar(20),
        PRIMARY key(buyerID, PaymentId),
        FOREIGN key(buyerID) REFERENCES buyer(buyerID)
```

```
)
CREATE TABLE HISTORY(
        HID int(10),
        userID varchar(50),
        AddedDate date,
        PRIMARY key(HID),
        FOREIGN key(userID) REFERENCES Buyer(buyerID) ON DELETE CASCADE
)
CREATE TABLE CARRIER(
        CarrierPhone  INT,
        cname varchar(10),
        cemail varchar(30),
        CarrierID INT,
        PRIMARY key(CarrierID)
)
ALTER TABLE `carrier`
CHANGE `CarrierID` INT NOT NULL AUTO_INCREMENT; CREATE
TABLE CART(
        CID INT,
        UserID varchar(50),
        AddedDate Date,
        ItemCount INT,
        PRIMARY key(CID),
        FOREIGN key(UserID) REFERENCES Buyer(BuyerID) on DELETE CASCADE
)
ALTER TABLE `cart` CHANGE `CID` `CID` INT NOT NULL AUTO_INCREMENT;
CREATE TABLE CHECKOUT(
        CHKID INT,
        CID INT,
        PRIMARY key(CHKID),
        FOREIGN key(CID) REFERENCES cart(CID) on DELETE CASCADE
)
```

```
CREATE TABLE PROCEED_ORDER(
	PaymentID INT,
	userID varchar(50),
	CHKID INT,
	CID INT,
	PRIMARY KEY(PaymentID,UserID, CHKID, CID),
	 FOREIGN key(CID) REFERENCES cart(CID) on DELETE CASCADE,
	FOREIGN key(CHKID) REFERENCES checkout(CHKID) on DELETE CASCADE
)
ALTER TABLE ` PROCEED_ORDER ` ADD CONSTRAINT `proceed_order_ibfk_3`
FOREIGN KEY (`UserID`) REFERENCES `PaymentInfo`(`BuyerID`) ON DELETE
RESTRICT ON UPDATE RESTRICT;

CREATE TABLE ORDERDETAILS(
	OrderID INT,
	Address varchar(10),
	ItemName varchar(20),
	Quantity INT,
	Cost INT,
	Order_Date Date,
	UserID varchar(50),
	HID INT,
	CID INT,
	CarrierID INT,
	Paystatus varchar(10),
	PRIMARY key(OrderID),
	FOREIGN key(UserID) REFERENCES buyer(BuyerID) on DELETE CASCADE,
	FOREIGN key(HID) REFERENCES history(HID) on DELETE CASCADE,
	FOREIGN key(CID) REFERENCES cart(CID) on DELETE CASCADE,
	FOREIGN key(CarrierID) REFERENCES carrier(CarrierID) on DELETE CASCADE
)
```

```
CREATE TABLE PRODUCTORDERED(
        OrderID INT,
        ItemID INT,
        PRIMARY key(OrderID, ItemID),
        FOREIGN key(OrderID) REFERENCES order_details(OrderID) on DELETE
        CASCADE, FOREIGN key(ItemID) REFERENCES product(ItemID) on DELETE
        CASCADE
)
CREATE TABLE REVIEW(
        ReviewID INT NOT NULL AUTO_INCREMENT,
        Comments varchar(30),
        Date Date,
        Reviewtype varchar(10),
        PRIMARY
        key(ReviewID)
)
CREATE TABLE REVIEW_IMAGE(
        ReviewID INT,
        Id INT NOT NULL AUTO_INCREMENT,
        Image  varchar(30),
        Added_Date   Date,
        PRIMARY key(id),
        FOREIGN key(reviewID) REFERENCES review(reviewID) on DELETE CASCADE
)
CREATE TABLE PROVIDE_REVIEW(
        ReviewID INT,
        userID varchar(10),
        ItemID INT,
        PRIMARY key(ReviewID,userID,ItemID),
        FOREIGN key(ReviewID) REFERENCES review(ReviewID) on DELETE
        CASCADE, FOREIGN key(UserID) REFERENCES buyer(BuyerID) on DELETE
        CASCADE, FOREIGN key(ItemID) REFERENCES product(ItemID) on DELETE
        CASCADE
)
CREATE TABLE WISHLIST(
        WID INT AUTO_INCREMENT ,
        UserID varchar(30),
        Date Date,
        PRIMARY key(WID),
        FOREIGN key(UserID) REFERENCES Buyer(BuyerID) on DELETE CASCADE
)

CREATE TABLE CART_DISPLAYS(
        CID INT,
        ItemID INT,
        PRIMARY key(CID,ItemID),
        FOREIGN key(ItemID) REFERENCES Product(ItemID) on DELETE CASCADE
)

ALTER TABLE ` CART_DISPLAYS ` DROP FOREIGN KEY `
CART_DISPLAYS _ibfk_1`; ALTER TABLE ` CART_DISPLAYS ` ADD
CONSTRAINT ` CART_DISPLAYS _ibfk_1`

FOREIGN KEY (`ItemID`) REFERENCES `PRODUCT`(`ItemID`) ON DELETE CASCADE
ON UPDATE RESTRICT;

ALTER TABLE ` CART_DISPLAYS ` ADD CONSTRAINT ` CART_DISPLAYS _ibfk_2`
FOREIGN KEY (`CID`) REFERENCES `cart`(`CID`) ON DELETE CASCADE ON
UPDATE RESTRICT;

CREATE TABLE WISHLIST_CONTAINS(
        WID INT,
        ItemID INT,
        PRIMARY key(WID, ItemID),
        FOREIGN key(WID) REFERENCES wishlist(WID) on DELETE CASCADE,
        FOREIGN key(ItemID) REFERENCES product(ItemID) on DELETE CASCADE
)
```

PL/SQL

PROCEDURES

1. Buyer_ Registration:

This Procedure is invoked from Update_USERS Trigger when a

user is inserted.

Usecase:

- If the User is a Buyer, create a record in the Buyer table. The user will be enrolled as a Buyer by default.

2. Seller_ Registration:

This Procedure is invoked from Update_USERS Trigger when a

user is inserted.

Usecase:

- If the User is a Seller, create a record in the Seller table

3. ClearCart

This Procedure is invoked from CLEAR_CART Trigger when a checkout is executed.

Usecase:

- Whenever a new checkout is executed, the items in the cart are cleared.

4. Add_Product

This Procedure is invoked from Populate_Products Trigger.

Usecase:

- Insert records into the category table based on the product category.
- Insert a record in either BID or DIRECTBUY depending on the type of product.
- If the item belongs to BID, the AuctionOffer table will be updated based on the offer code.

5. Update_History

This Procedure is invoked from HISTORY_UPDATE Trigger when an order is finished.

Usecase:

- Whenever a new order is completed, the history of that order is updated in the History Table.

TRIGGERS

1. Update_USERS:

- When a new user is created, the stored procedure responsible for adding records into the Buyer and Seller tables is called.

2. CLEAR_CART:

- After successful checkout, the products in the cart are cleared by invoking ClearCart procedure.

3. Populate_Products:

- Invokes the Add_Products procedure which refreshes the Items, Bid and DirectBuy Table.

4. History_Update:

- The Update_History procedure is called when this trigger is executed which helps us to save the history of the transaction after order is checked out.

5. Check_Product_Instock:

- Before insert on Orderdetails this trigger is invoked to check if the item is in stock to purchase and throws an error if it is not available.

PL/SQL STORED PROCEDURES

Procedure-1 Buyer_ Registration :

```
CREATE OR REPLACE PROCEDURE BUYER_REGISTRATION (email in USER.EmailID
%Type) AS
BEGIN
        Insert into BUYER VALUES (
                email
        );
END BUYER_REGISTRATION;
```

Procedure-2 Seller_ Registration :

```
CREATE OR REPLACE PROCEDURE SELLER_ REGISTRATION (
        email in USER.EmailID %Type,
        company_name in SELLER.Company_name%Type,
        description in SELLER.Description%Type,
        rating in SELLER.Rating%Type,
) AS
BEGIN
        Insert into SELLER VALUES (
                email,
                company_name,
                description,
                rating
        );
END SELLER_ REGISTRATION;
```

Procedure-3 ClearCart

```
CREATE OR REPLACE PROCEDURE CLEARCART (cid in CART.CID %Type) AS
```

```
BEGIN
DELETE FROM CART WHERE CID=cid;
)
END CLEARCART;
```

Procedure-4 Add_Product

```
CREATE OR REPLACE PROCEDURE ADD_PRODUCT (
          Price    Product.price%TYPE,
          Image Product.image%TYPE
          ItemNameProduct.ItemName%TYPE
       Description Product.Description%TYPE,
       CATID INT,
       ITEMTYPE       VARCHAR,
       OfferCode      VARCHAR,
)AS
ID        Product.ItemID%TYPE
Discount Auction_Offer.Discount%TYPE
BEGIN
INSERT INTO Product VALUES (
          Price,
          Image,
          Title,
          Description
)
SELECT ItemID INTO ID FROM Items WHERE ItemID = :NEW.ItemID
INSERT INTO CATEGORY VALUES (
          CATID,
          CName,
)
```

```
IF (ITEMTYPE=0) THEN
        INSERT INTO BID VALUES (
                ItemID, Price, StartDate, EndDate
        )
        INSERT INTO AUCTION_OFFER VALUES (
                OfferCode,
                Discount,
                ItemID,
                BidID,
        )
ELSE

                INSERT     INTO DIRECTBUY VALUES ( ItemID, Price, BuyID
)
END IF;
END ADD_PRODUCT;
```

Procedure-5 Update History

```
CREATE OR REPLACE PROCEDURE INSERT_HISTORY(ORDER_ID in
ORDER_DETAILS.OrderID%TYPE, UserID in ORDER_DETAILS.UserId%TYPE, OrderedDate in
ORDER_DETAILS.Order_Date%TYPE) AS
BEGIN
INSERT INTO HISTORY VALUES (ORDER_ID,userID,orderedDate)
END INSERT_HISTORY;
```

PL/SQL-TRIGGERS

Trigger-1 Update_USERS :

```
CREATE OR REPLACE TRIGGER UPDATE_USERS AFTER INSERT
ON USERS
FOR EACH ROW
DECLARE
        seller          USERS.isSeller %Type
BEGIN
       Select isSeller into seller from users where email = :NEW.EmailID;
       IF (seller =’Y’) THEN
              BUYER_REGISTRATION (NEW.EmailID);
       ELSE
              SELLER_REGISTRATION (NEW.EmailID,”Zexa Tech Ltd”,”Power”,4)
      END IF;
END;
```

Trigger-2 CLEAR_CART :

```
CREATE OR REPLACE TRIGGER CLEAR_CART
AFTER INSERT ON CHECKOUT
FOR EACH ROW
DECLARE
          cid CHECKOUT.CID%TYPE;
BEGIN
          select CID into cid from CHECKOUT where CHKID=: NEW.CHKID;
          CLEARCART(cid);
END;
```

Trigger-3 Populate_Products :

```
CREATE OR REPLACE TRIGGER POPULATE_PRODUCTS AFTER INSERT ON ITEM
FOR EACH ROW
BEGIN
        ADD_PRODUCT (23145,240,'nike.com/aexcv.jpg','Nike Jordan Air Max','Premium BasketBall
  Shoes','nikhila.s@gmail.com','Shoes',5,'5346','Y');
        ADD_PRODUCT (93456,45,'amazon.com/vfsvb.png','Alexa Echo Dot','Smart
  Speaker','harshini.s@gmail.com','Speaker',4,'2869','N');
END;
```

Trigger-4 History_Update

```
CREATE OR REPLACE TRIGGER HISTORY_ UPDATE
AFTER INSERT ON ORDER DETAILS
FOR EACH ROW
DECLARE
        orderdate    ORDERDETAILS.Order_Date%TYPE;
        userid       ORDER_DETAILS.UserId%TYPE;
BEGIN
  select UserId,Order_Date into userid,orderdate from ORDER_DETAILS where OrderId=: NEW.OrderId;
  INSERT_HISTORY(orderid,userid,orderdate);
END;
```

Trigger-5 Check_Product_Instock

```
CREATE OR REPLACE TRIGGER CHECK_PRODUCT_INSTOCK
BEFORE INSERT ON ORDERDETAILS
FOR EACH ROW
BEGIN

    Select ItemID from PRODUCT where ItemID = :NEW.ItemID;
    IF (inStock = 'N') THEN
       raise_application_error( -20111, 'The Item is not in stock to order');
    END IF;
END;
```