E-bay

E-commerce Database System

Project Report

Team No: Team-1, Ebay-1

Team Members:

Shanjida Khatun (sxk200130@utdallas.edu)
Saif Mahmood (sxm190136@utdallas.edu)
Ajey Bangalore Subrahmanya (axb200054@utdallas.edu)

Google Drive Link:
https://drive.google.com/drive/folders/1KS1bAz2NKICAT3Yv0Xt2oKxJGsoLAdXM?usp=sharing

For CS6360.003
Submission Date: 12/02/2020
Professor: Dr. Nurcan Yuruk

Table of Contents

Section 1: Requirements

Consider a database system for an e-commerce website (e.g., eBay) that facilitates online selling and buying of products. The data requirements for this system are summarized as follows:

1. The e-commerce website offers membership which is uniquely identified by its username and/or email address. Each member account also has additional information such as contact and payment information, account type, and sign-in security details.
2. A member can be a seller, a buyer, or both. A seller account has shipping from address and credit merchant account (e.g., VISA) number for receiving payments. A buyer account has a shipping address for billing and shipping purposes.
3. A seller can list items for sale in either an “auction” or “buy-now”. To sell the same items in both auction and buy-now, a seller has to list them separately. Each listing contains details of the item’s description, condition, location, and available quantity. It also has the item category and type of shipping offered.
4. In a buy-now listing, buyers buy items directly based on the price set in the listing. A buyer can set item quantity in its purchase. The number of items sold and available product quantity is tracked for buy-now listings.
5. In an auction listing, the seller sets the reserve price (i.e., the minimum amount the seller is willing to sell the item for), auction end time, and bid increment value. The winning bid and winning buyer (i.e., the buyer with the winning bid) are determined at the end of the auction. The product quantity is fixed for an auction. A seller must create multiple auction listings with unique IDs to sell the same item multiple times.
6. A seller cannot buy or bid on its own listed items.
7. At the end of each item purchase, either through buy-now or auction, a unique order number is created that identifies the corresponding buyer and seller, order time, expected and actual shipping dates, the total amount to be paid, and seller fees.
8. Comments and ratings for both buyer and seller are recorded for every order. Each member’s feedback score is calculated based on the ratings (from buyer and seller) from its orders.

Section 2: EER Diagram

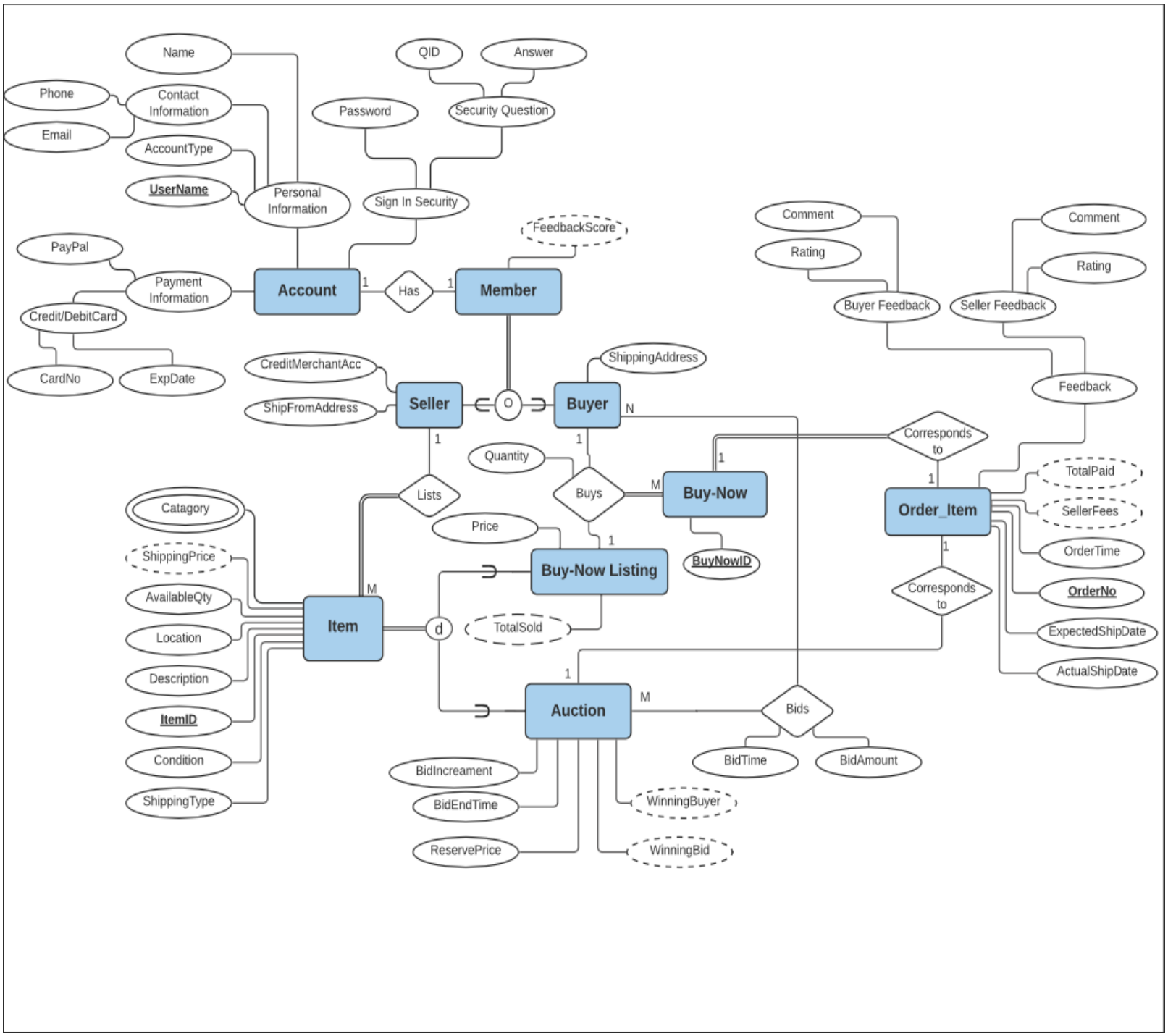


Figure 1: Initial EER diagram

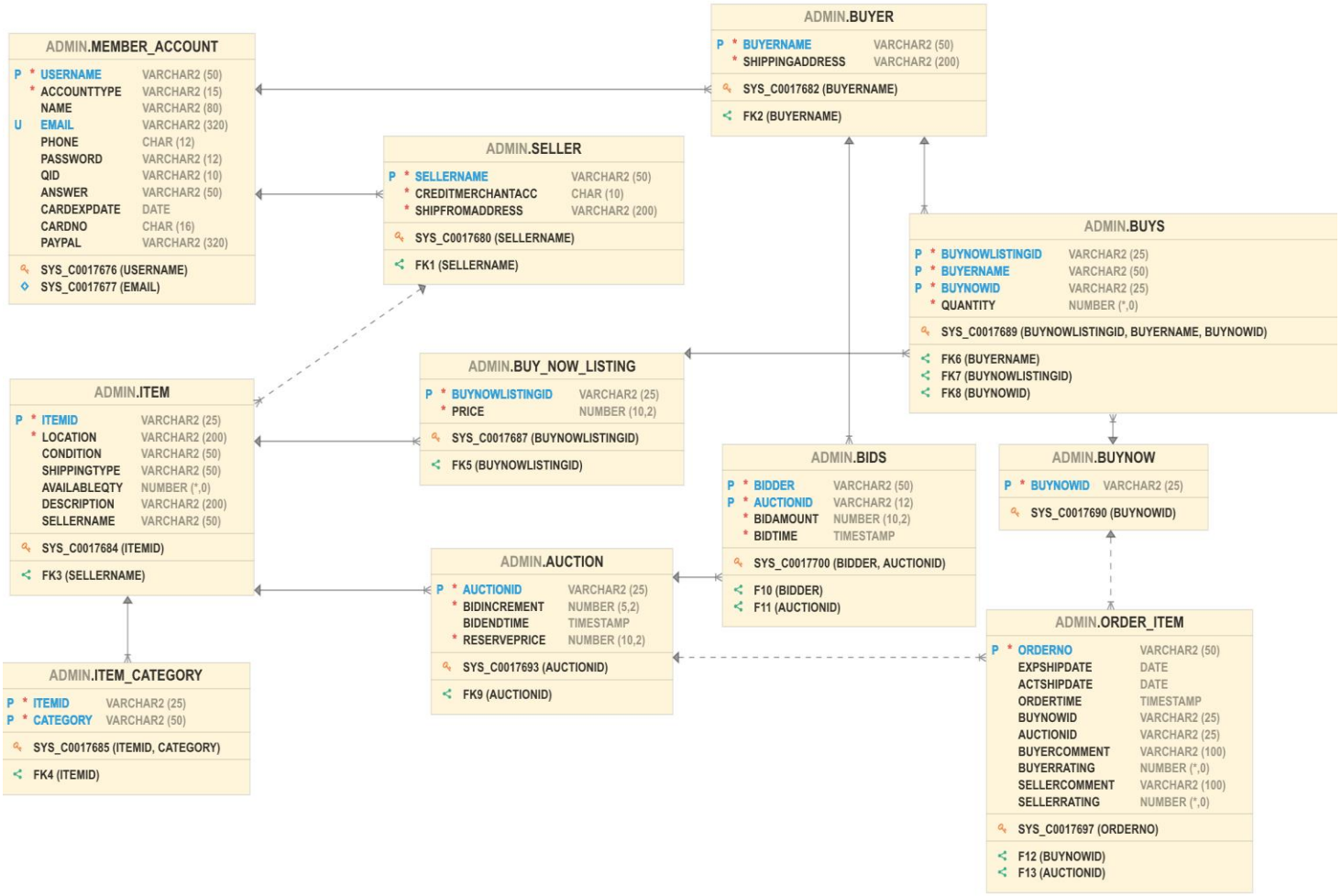


Figure 2: Final EER diagram

Section 3: Relational Schema

Member_Account

UserName	AccountType	Name	Email	Phone	Password	QID	Answer	CardExpDate	CardNo	PayPal

Seller

SellerName(FK)	CreditMerchantAcc	ShipFromAddress

Buyer

BuyerName(FK)	ShippingAddress

Item

ItemID	Location	Condition	ShippingType	AvailableQty	Description	SellerName(FK)

Item_Category

ItemID(FK)	Category

Buy_Now_Listing

BuyNowListingID(FK)	Price

Buys

BuyNowListingID(FK)	BuyerName(FK)	BuyNowID(FK)	Quantity

BuyNow

BuyNowID

Auction

AuctionID(FK)	BidIncrement	BidEndTime	ReservePrice

Bids

Bidder(FK)	AuctionID(FK)	BidAmount	BidTime

Order_Item

OrderNo	ExpShipDate	ActShipDate	OrderTime	BuyNowID(FK)	AuctionID(FK)	BuyerComment	BuyerRating	SellerComment	SellerRating

Referential Integrity Constraints

- TABLE Seller has FOREIGN KEY(SellerName) that REFERENCES Member_Account(UserName)
- TABLE Buyer has FOREIGN KEY(BuyerName) that REFERENCES Member_Account(UserName)
- TABLE Item has FOREIGN KEY(SellerName) that REFERENCES Seller(SellerName)
- TABLE Item_Category has FOREIGN KEY(ItemID) that REFERENCES Item(ItemID)
- TABLE Buy_Now_Listing has FOREIGN KEY(BuyNowListingID) that REFERENCES Item(ItemID)
- TABLE Auction has FOREIGN KEY(AuctionID) that REFERENCES Item(ItemID)
- TABLE Bids has FOREIGN KEY(Bidder) that REFERENCES Buyer(BuyerName)
- TABLE Bids has FOREIGN KEY(AuctionID) that REFERENCES Auction(AuctionID)
- TABLE Order_Item has FOREIGN KEY(BuyNowID) that REFERENCES BuyNow(BuyNowID)
- TABLE Order_Item has FOREIGN KEY(AuctionID) that REFERENCES Auction(AuctionID)
- TABLE Buys has FOREIGN KEY(BuyNowListingID) that REFERENCES Buy_Now_Listing(BuyNowListingID)
- TABLE Buys has FOREIGN KEY(BuyerName) that REFERENCES Buyer(BuyerName)
- TABLE Buys has FOREIGN KEY(BuyNowID) that REFERENCES BuyNow(BuyNowID)

Section 4: Normalization

Member_Account: It obeys all 1st, 2nd,3rd Normalization forms.
There is only one key 'UserName' in Member_Account.
All the column data 'AccountType, Name, Email, Phone, Password, QID, Answer, CardExpDate, CardNo, PayPal' depends upon only the key 'UserName'.

The 1st Normal form is obeyed since there is no duplicate data.
The 2nd Normal form is obeyed since the column data 'AccountType, Name, Email, Phone, Password, QID, Answer, CardExpDate, CardNo, PayPal' only depends upon the whole key value which is 'UserName'.
The 3rd Normal form is obeyed as there is no column data that depends on any other data other than the whole key or the composite key.

Seller: It obeys all 1st, 2nd,3rd Normalization forms.
Seller has only one key 'SellerName'. The column data 'CreditMerchantAcc, ShipFromAddress' depends upon only the key.

The 1st Normal form is obeyed since there is no duplicate data.
The 2nd Normal form is obeyed since the column data 'CreditMerchantAcc, ShipFromAddress' only depends upon the whole key value which is 'SellerName'.
The 3rd Normal form is obeyed as there is no column data that depends on any other data other than the key.

Buyer: It obeys all 1st, 2nd,3rd Normalization forms.
Buyer has only one key 'BuyerName'. The column data 'ShippingAddress' depends upon only the key.

The 1st Normal form is obeyed since there is no duplicate data.
The 2nd Normal form is obeyed since the column data 'ShippingAddress' only depends upon the whole key value which is 'BuyerName'.
The 3rd Normal form is obeyed as there is no column data that depends on any other data other than the key.

Item: It obeys all 1st, 2nd,3rd Normalization forms.
There is only one key 'ItemID' in Item. All the column data 'Location, Condition, ShippingType, AvailableQty, Description, SellerName' depends upon only the key 'ItemID'.

The 1st Normal form is obeyed since there is no duplicate data.
The 2nd Normal form is obeyed since the column data 'Location, Condition, ShippingType, AvailableQty, Description, SellerName' only depends upon the whole key value which is 'ItemID'.
The 3rd Normal form is obeyed as there is no column data that depends on any other data other than the whole key or the composite key.

Item_Category: It obeys all 1st, 2nd,3rd Normalization forms.
Item_Category has a composite key 'ItemID, Category'.

The 1st Normal form is obeyed since there is no duplicate data.
The 2nd Normal form is obeyed since both the column data are key.
The 3rd Normal form is obeyed as there is no column data that depends on any other data other than the whole key or the composite key.

Buy_Now_Listing: It obeys all 1st, 2nd,3rd Normalization forms.
The column data 'BuyNowListingID' in Buy_Now_Listing is the key.

The 1st Normal form is obeyed since there is no duplicate data.
The 2nd Normal form is obeyed since the column data 'Price' only depends upon the key value which is 'Buy_Now_Listing'.
The 3rd Normal form is obeyed as there is no column data that depends on any other data other than the whole key or the composite key.

Buys: It obeys all 1st, 2nd,3rd Normalization forms.
Buys has a composite key 'BuyNowListingID, BuyerName, BuyNowID'. The column data 'Quantity' depends upon only the key values.

The 1st Normal form is obeyed since there is no duplicate data.
The 2nd Normal form is obeyed since the column data 'Quantity' only depends upon the whole key value which is 'BuyNowListingID, BuyerName, BuyNowID'.
The 3rd Normal form is obeyed as there is no column data that depends on any other data other than the key.

BuyNow: It obeys all 1st, 2nd,3rd Normalization forms.
BuyNow has only one key 'BuyNowID'. There is no column data other than the key.

The 1st Normal form is obeyed since there is no duplicate data.
The 2nd Normal form is obeyed since the only column data is also the key.
The 3rd Normal form is obeyed as there is no column data that depends on any other data other than the key.

Auction: It obeys all 1st, 2nd,3rd Normalization forms.
Auction has only one key 'AuctionID'. The column data 'BidIncrement, BidEndTime, ReservePrice' depends upon only the key.

The 1st Normal form is obeyed since there is no duplicate data.
The 2nd Normal form is obeyed since the column data 'BidIncrement, BidEndTime, ReservePrice' only depends upon the whole key value which is 'AuctionID'.

The 3rd Normal form is obeyed as there is no column data that depends on any other data other than the key.

Bids: It obeys all 1st, 2nd,3rd Normalization forms.
Bids has a composite key 'Bidder, AuctionID'. The column data 'BidAmount and BidTime' depends upon only the key.

The 1st Normal form is obeyed since there is no duplicate data.
The 2nd Normal form is obeyed since the column data 'BidAmount and BidTime' depends upon the whole key value which is 'Bidder' and 'AuctionID'.
The 3rd Normal form is obeyed as there is no column data that depends on any other data other than the key.

Order_item: It obeys all 1st, 2nd,3rd Normalization forms.
Order_Item has a composite key 'OrderNo, AuctionID, BuyitNowID'. The column data 'ExpShipDate, ActShipDate, OrderTime, BuyerComment, BuyerRating, SellerComment, SellerRating' depends upon the keys.

The 1st Normal form is obeyed since there is no duplicate data.
The 2nd Normal form is obeyed since the column data 'ExpShipDate, ActShipDate, OrderTime, BuyerComment, BuyerRating, SellerComment, SellerRating' depends upon the whole key value which is 'OrderNo, AuctionID, BuyitNowID'.
The 3rd Normal form is obeyed as there is no column data that depends on any other data other than the key.

Section 5: Relational Schema after Normalization

Since there was no violation for 1NF, 2NF & 3NF found, therefore, the Relational Schema remains the same as before.

Section 6: Table Insertion using SQL commands

```
/* Part-1: Create table statements */

CREATE TABLE Member_Account (
  UserName        varchar(50),
  AccountType     varchar(15) not null,
  Name            varchar(80),
  Email           varchar(320),
  Phone           char(12),
  Password        varchar(12),
  QID             varchar(10),
  Answer          varchar(50),
```

```
  CardExpDate     date,
  CardNo          char(16),
  PayPal          varchar(320),
  primary key (UserName),
  unique (Email)
);

CREATE TABLE Seller (
  SellerName        varchar(50),
  CreditMerchantAcc char(10) not null,
  ShipFromAddress   varchar(200) not null,
  primary key (SellerName )
);

CREATE TABLE Buyer (
  BuyerName         varchar(50),
  ShippingAddress   varchar(200) not null,
  primary key (BuyerName )
);

CREATE TABLE Item (
  ItemID                varchar(25),
  Location              varchar(200) not null,
  Condition             varchar(50),
  ShippingType          varchar(50),
  AvailableQty          integer,
  Description           varchar(200),
  SellerName            varchar(50),
  primary key (ItemID)
);

CREATE TABLE Item_Category (
  ItemID     varchar(25),
  Category   varchar(50),
  primary key (ItemID,Category)
);

CREATE TABLE Buy_Now_Listing (
  BuyNowListingID     varchar(25),
  Price               decimal(10,2) not null,
  primary key (BuyNowListingID)
);

CREATE TABLE Buys (
  BuyNowListingID varchar(25),
  BuyerName       varchar(50),
  BuyNowID        varchar(25),
  Quantity        integer not null,
  primary key (BuyNowListingID,BuyerName,BuyNowID)
);

CREATE TABLE BuyNow (
  BuyNowID        varchar(25),
  primary key (BuyNowID)
);
```

```
CREATE TABLE Auction (
  AuctionID      varchar(25),
  BidIncrement   decimal(5,2) not null,
  BidEndTime     timestamp,
  ReservePrice   decimal(10,2) not null,
  primary key (AuctionID)
);

CREATE TABLE Bids (
  Bidder          varchar(50),
  AuctionID       varchar(25),
  BidAmount       decimal(10,2) not null,
  BidTime         timestamp not null,
  primary key (Bidder, AuctionID )
);

CREATE TABLE Order_Item (
  OrderNo          varchar(50),
  ExpShipDate      date,
  ActShipDate      date,
  OrderTime        timestamp,
  BuyNowID         varchar(25),
  AuctionID        varchar(25),
  BuyerComment     varchar(100),
  BuyerRating      integer,
  SellerComment    varchar(100),
  SellerRating     integer,
  primary key (OrderNo )
);

/* Part-2: Foreign Keys and Triggers */

ALTER TABLE Seller ADD CONSTRAINT fk1 FOREIGN KEY(SellerName) REFERENCES
Member_Account(UserName) ON DELETE SET NULL;

ALTER TABLE Buyer ADD CONSTRAINT fk2 FOREIGN KEY(BuyerName)REFERENCES
Member_Account(UserName)ON DELETE SET NULL;

ALTER TABLE Item ADD CONSTRAINT fk3 FOREIGN KEY(SellerName)REFERENCES
Seller(SellerName)ON DELETE SET NULL;

ALTER TABLE Item_Category ADD CONSTRAINT fk4 FOREIGN KEY(ItemID)REFERENCES
Item(ItemID)ON DELETE CASCADE;

ALTER TABLE Buy_Now_Listing  ADD CONSTRAINT fk5 FOREIGN
KEY(BuyNowListingID)REFERENCES Item(ItemID) ON DELETE SET NULL;

ALTER TABLE Auction ADD CONSTRAINT fk9 FOREIGN KEY(AuctionID)REFERENCES
Item(ItemID)ON DELETE SET NULL;

ALTER TABLE Buys ADD CONSTRAINT fk6 FOREIGN KEY(BuyerName)REFERENCES
Buyer(BuyerName)ON DELETE CASCADE;

ALTER TABLE Buys ADD CONSTRAINT fk7 FOREIGN KEY(BuyNowListingID)REFERENCES
Buy_Now_Listing(BuyNowListingID)  ON DELETE CASCADE;
```

```
ALTER TABLE Buys ADD CONSTRAINT fk8 FOREIGN KEY(BuyNowID)REFERENCES
BuyNow(BuyNowID)ON DELETE CASCADE;

ALTER TABLE Bids ADD CONSTRAINT f10 FOREIGN KEY(Bidder)REFERENCES
Buyer(BuyerName)ON DELETE CASCADE;

ALTER TABLE Bids ADD CONSTRAINT f11 FOREIGN KEY(AuctionID)REFERENCES
Auction(AuctionID)ON DELETE CASCADE;

ALTER TABLE Order_Item ADD CONSTRAINT f12 FOREIGN KEY(BuyNowID)REFERENCES
BuyNow(BuyNowID)ON DELETE SET NULL;

ALTER TABLE Order_Item ADD CONSTRAINT f13 FOREIGN KEY(AuctionID)REFERENCES
Auction(AuctionID)ON DELETE SET NULL;

/* In case, need to start-over */

drop table member_account;
drop table seller;
drop table buyer;
drop table item;
drop table item_category;
drop table Buy_Now_Listing;
drop table auction;
drop table bids;
drop table buys;
drop table buynow;
drop table order_item;

/* In case, need to check the table values */

select * from member_account;
select * from seller;
select * from buyer;
select * from item;
select * from item_category;
select * from Buy_Now_Listing;
select * from auction;
select * from bids;
select * from buys;
select * from buynow;
select * from order_item;

/* In case, need to create the tables for apex application */

Create table projectuser.member_account as (select * from
admin.member_account);
Create table projectuser.item as (select * from admin.item);
Create table projectuser.buyer as (select * from admin.buyer);
Create table projectuser.seller as (select * from admin.seller);
Create table projectuser.item_category as (select * from
admin.item_category);
Create table projectuser.buy_now_listing as (select * from
admin.buy_now_listing);
Create table projectuser.buynow as (select * from admin.buynow);
Create table projectuser.buys as (select * from admin.buys);
```

```
Create table projectuser.bids as (select * from admin.bids);
Create table projectuser.order_item as (select * from admin.order_item);
Create table projectuser.auction as (select * from admin.auction);

/* In case, need to start-over for apex application*/

drop table projectuser.member_account;
drop table projectuser.item;
drop table projectuser.buyer;
drop table projectuser.seller;
drop table projectuser.item_category;
drop table projectuser.buy_now_listing;
drop table projectuser.buynow;
drop table projectuser.buys;
drop table projectuser.bids;
drop table projectuser.order_item;
drop table projectuser.auction;
```

Section 7: Procedures and Triggers

Procedures:

```
/*A Stored Procedure that will calculate and display the winning buyer and
winner bid  for that particular item and will store the all information in
the log file for each item.*/

CREATE TABLE WinBid_Log (
ItemID            VARCHAR(25),
Winning_Buyer     varchar(50),
Winning_Bid       decimal(10,2)
);

create or replace PROCEDURE Display_Win_Bids AS

thisItem Bids%ROWTYPE;
CURSOR AuctBids IS
 Select * from bids where (auctionid, bidamount) in (SELECT b.auctionid,
max(b.bidamount) FROM Bids B  group by b.auctionid);

BEGIN
OPEN AuctBids;
LOOP
  FETCH AuctBids INTO thisItem;
  EXIT WHEN (AuctBids%NOTFOUND);
  INSERT INTO WinBid_Log VALUES (thisItem.AuctionID, thisItem.Bidder,
thisItem.BidAmount);

  dbms_output.put_line('Item_ID:  '||thisItem.AuctionID);
  dbms_output.put_line('Winning Buyer:  '||thisItem.Bidder);
  dbms_output.put_line('Winner Bid:  '||thisItem.BidAmount);

END LOOP;
CLOSE AuctBids;
END;
```

```
begin
Display_Win_Bids();
end;

/*For checking*/
select * from WinBid_Log;
drop table WinBid_Log;


/* A Procedure that will update the available item quantity for a particular
item when a seller wants to add additional items.*/

create or replace PROCEDURE Item_Increase(itemnumber IN item.itemid%TYPE, qty
IN integer) AS

thisItem item%ROWTYPE;

CURSOR Dept5Emps IS
SELECT i.* FROM item i WHERE i.itemid=itemnumber
FOR UPDATE;

BEGIN
OPEN Dept5Emps;
LOOP
  FETCH Dept5Emps INTO thisItem;
  EXIT WHEN (Dept5Emps%NOTFOUND);
  dbms_output.put_line(thisItem.itemid);
  dbms_output.put_line(thisItem.availableqty);
  UPDATE item SET availableqty = availableqty + qty
  WHERE itemid= thisItem.itemid;

END LOOP;
CLOSE Dept5Emps;
END;

begin
Item_Increase('157334185326',1);
end;


/* A Stored Function that receives a seller username as input and calculates
the average rating scale given to that seller */

create or replace PROCEDURE Seller_Rating(sellername IN
Seller.Sellername%TYPE) AS

Rate number;
BEGIN

select avg(o.sellerrating) into rate from order_item o, buys b, item i where
i.itemid=b.buynowlistingid and b.buynowid=o.buynowid and
i.sellername=sellername;
dbms_output.put_line('Average Rating = ' || Rate);
END;
```

```
begin
Seller_Rating('bhumipatr');
end;

/*A Procedure that receives an item number as input, and displays all bids
(i.e. auction number, bidder number, bid amount, bid date and time) for that
particular item.*/

create or replace PROCEDURE Display_All_Bids(itemnumber IN
Bids.AuctionID%TYPE) AS

thisItem Bids%ROWTYPE;

CURSOR AuctBids IS
SELECT B.* FROM Bids B WHERE B.AuctionID=itemnumber;

BEGIN
OPEN AuctBids;
LOOP
  FETCH AuctBids INTO thisItem;
  EXIT WHEN (AuctBids%NOTFOUND);
  dbms_output.put_line(thisItem.Bidder || '          '|| thisItem.AuctionID
||'     '||  thisItem.BidAmount ||' '|| thisItem.BidTime);
END LOOP;
CLOSE AuctBids;
END;

begin
Display_All_Bids('153788526369');
end;
```

Triggers:

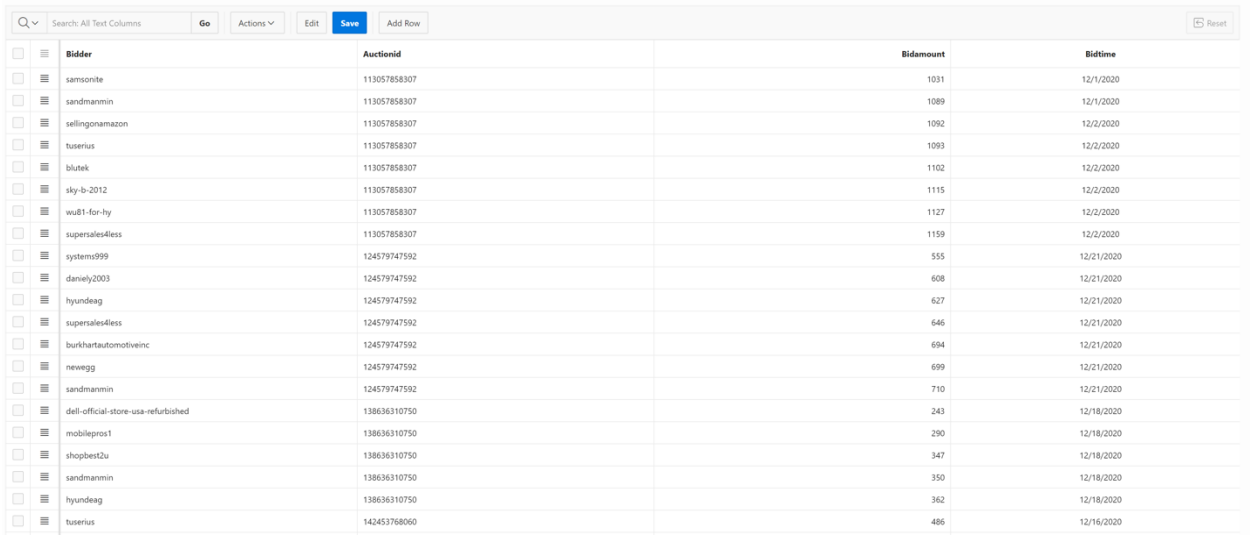
```
/*A Trigger which prevents any update or delete of Item details if there is
an existing bid on that item.*/

CREATE or REPLACE TRIGGER Item_details_changes
  BEFORE DELETE OR UPDATE  ON Item
  FOR EACH ROW
  DECLARE
  TYPE itemid_b            IS TABLE OF item.itemid%TYPE;
  bid_item                 itemid_b;

   BEGIN
      SELECT               auctionid
      BULK COLLECT INTO    bid_item
      FROM                 auction;

    FOR j IN 1..bid_item .COUNT() LOOP
       IF :NEW.itemid = bid_item(j)
       THEN
       Raise_Application_Error(-20000, 'Item exists on bid');
       END IF;
    END LOOP;
  END;
```

```
/*For checking*/
UPDATE item SET availableqty = availableqty + 1
WHERE itemid= '142453768060';


 /*A Trigger which prevents any quantity of buy-now Item that will greater
than the available quantity of the item in the store.*/


CREATE or REPLACE TRIGGER Buy_Quantity_changes
  BEFORE INSERT OR UPDATE  OF quantity ON BUYS
  FOR EACH ROW
  DECLARE
    itemquantity number;
BEGIN
    select availableqty into itemquantity from item where
itemid=:NEW.buynowlistingid;
    if itemquantity < :NEW.Quantity
      then Raise_Application_Error(-20000, 'Exceeded the available
quantity');
    end if;
END;

/*For checking*/
UPDATE BUYS SET quantity= 5000
WHERE buynowlistingid= '202452890737';


 /*A trigger that prevents a seller to buy or bid on its own listed items.*/

CREATE or REPLACE TRIGGER Seller_Bids
  BEFORE INSERT  ON Bids
  FOR EACH ROW
  DECLARE
    sellerid varchar(50);
BEGIN
    select sellername into sellerid from item where itemid=:NEW.auctionid;
    if sellerid < :NEW.bidder
      then Raise_Application_Error(-20000, 'Can not be a bidder');
    end if;
END;

CREATE or REPLACE TRIGGER Seller_Buy
  BEFORE INSERT  ON Buys
  FOR EACH ROW
  DECLARE
    sellerid varchar(50);
BEGIN
    select sellername into sellerid from item where
itemid=:NEW.buynowlistingid;
    if sellerid < :NEW.buyername
      then Raise_Application_Error(-20000, 'Can not be a buyer');
    end if;
END;
```

```
/*A trigger that prevents a bidder to  bid on an item after auction end
time.*/

CREATE or REPLACE TRIGGER Bidder_Bids
  BEFORE INSERT  ON Bids
  FOR EACH ROW
  DECLARE
    bidtime timestamp;
BEGIN
    select bidendtime into bidtime from auction where
auctionid=:NEW.auctionid;
    if bidtime < SYSDATE
      then Raise_Application_Error(-20000, 'Auction time ended');
    end if;
END;
```

Section 8: APEX Application

Auction Details



Bidder	Auctionid	Bidamount	Bidtime
samsonite	113057858307	1031	12/1/2020
sandmanmin	113057858307	1089	12/1/2020
sellingonamazon	113057858307	1092	12/2/2020
tuserius	113057858307	1093	12/2/2020
blutek	113057858307	1102	12/2/2020
sky-b-2012	113057858307	1115	12/2/2020
wu81-for-hy	113057858307	1127	12/2/2020
supersales4less	113057858307	1159	12/2/2020
systems999	124579747592	555	12/21/2020
daniely2003	124579747592	608	12/21/2020
hyundeag	124579747592	627	12/21/2020
supersales4less	124579747592	646	12/21/2020
burkhartautomotiveinc	124579747592	694	12/21/2020
newegg	124579747592	699	12/21/2020
sandmanmin	124579747592	710	12/21/2020
dell-official-store-usa-refurbished	138636310750	243	12/18/2020
mobilepros1	138636310750	290	12/18/2020
shopbest2u	138636310750	347	12/18/2020
sandmanmin	138636310750	350	12/18/2020
hyundeag	138636310750	362	12/18/2020
tuserius	142453768060	486	12/16/2020

Figure 3: Interactive Grid Report

Bids Summary

Itemid	Winning Buyer	Winning Bid
113057858307	supersales4less	1159
124579747592	sandmanmin	710
138636310750	hyundeag	362
142453768060	newegg	626
143018538922	supersales4less	576
153788526369	tuserius	1129
156926262853	dell-official-store-usa-refurbished	1126
157334185326	systems999	662
157745905795	tuserius	344
157997279939	supersales4less	1136
161665149377	mobilepros1	829
162763740503	sandmanmin	938
167814295325	sellingonamazon	676
168172621953	shopbest2u	450
168341386690	sandmanmin	303

1 - 15 Next

Figure 4: Classic Report

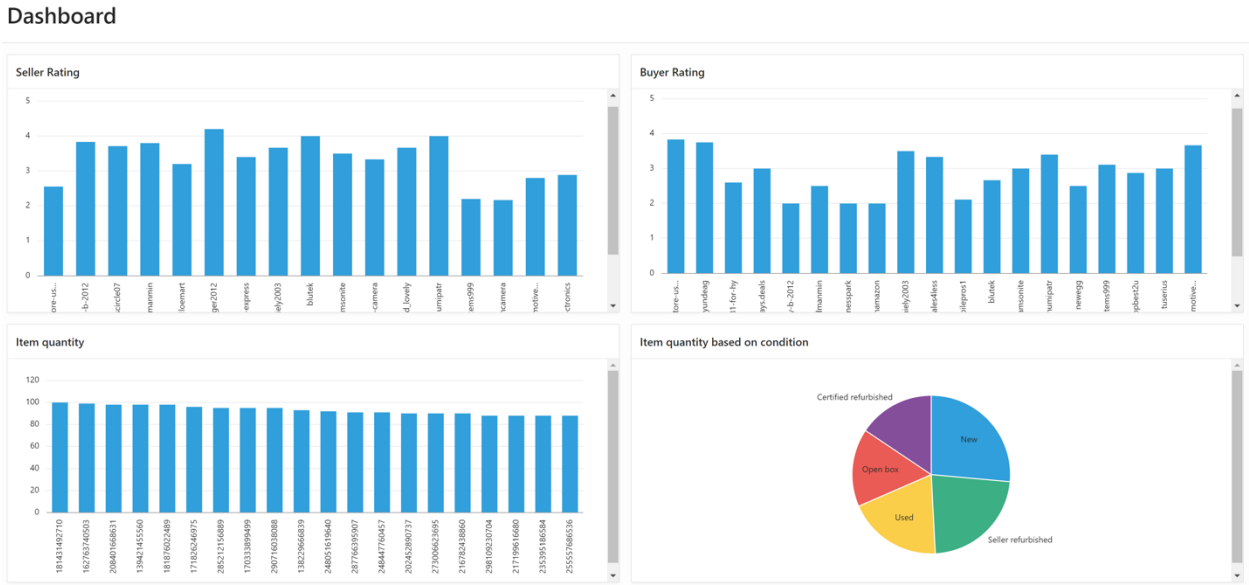


Figure 5: Dashboard

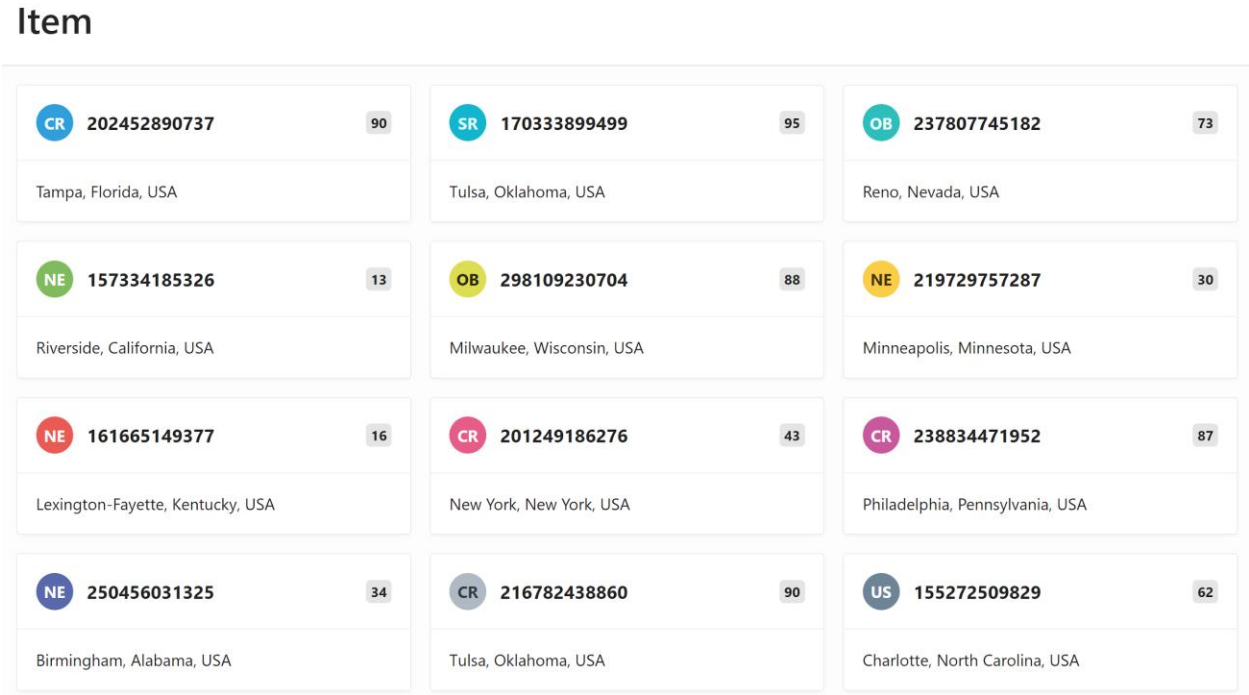


Figure 6: Item Cards

Order Summary

Go | Actions | Reset

Orderno	Sellername	Buyername	Total Paid
Order-582140	helloemart	nybusinesspark	58
Order-550128	daniely2003	tuserius	1827
Order-568826	daniely2003	systems999	2610
Order-539272	dell-official-store-usa-refurbished	sandmanmin	1074
Order-507745	dell-official-store-usa-refurbished	shopbest2u	3580
Order-578240	dell-official-store-usa-refurbished	mobilepros1	1432
Order-556061	dell-official-store-usa-refurbished	tuserius	1790
Order-544583	thewirelesscircle07	daniely2003	574
Order-572541	thewirelesscircle07	bhumipatr	3444
Order-582447	thewirelesscircle07	always.deals	574
Order-544462	samsonite	nybusinesspark	371
Order-505791	samsonite	mobilepros1	3339
Order-562727	systems999	mobilepros1	3630
Order-560876	systems999	samsonite	605
Order-512581	systems999	burkhartautomotiveinc	3025

Figure 7: Interactive Report

Item Shipping Date

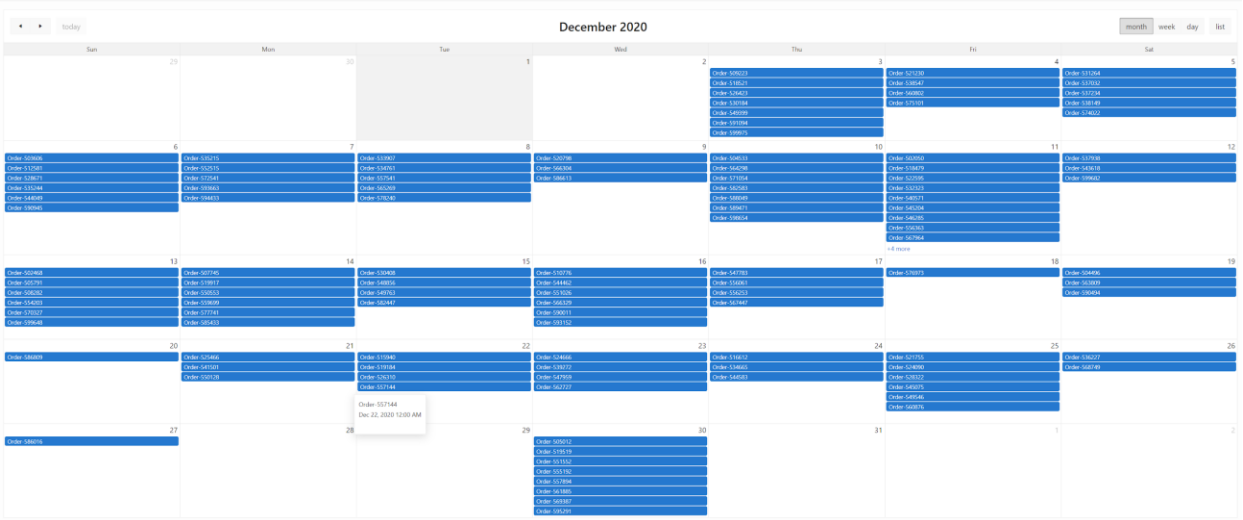


Figure 8:Calendar