

COMP0178: Database Fundamentals

Auction website coursework report

Group 3

Junming Hu

Jiaming Zhang

Arjun Bahra

Arei Mohammed

YouTube video demonstration:

https://www.youtube.com/watch?v=WBzHXtHd71o&ab\_channel=AreiM

December 2022

Table of Contents

[Introduction 3](#_Toc121976430)

[ER Diagram 3](#_Toc121976431)

[Database schema 4](#_Toc121976432)

[Database relationship table 5](#_Toc121976433)

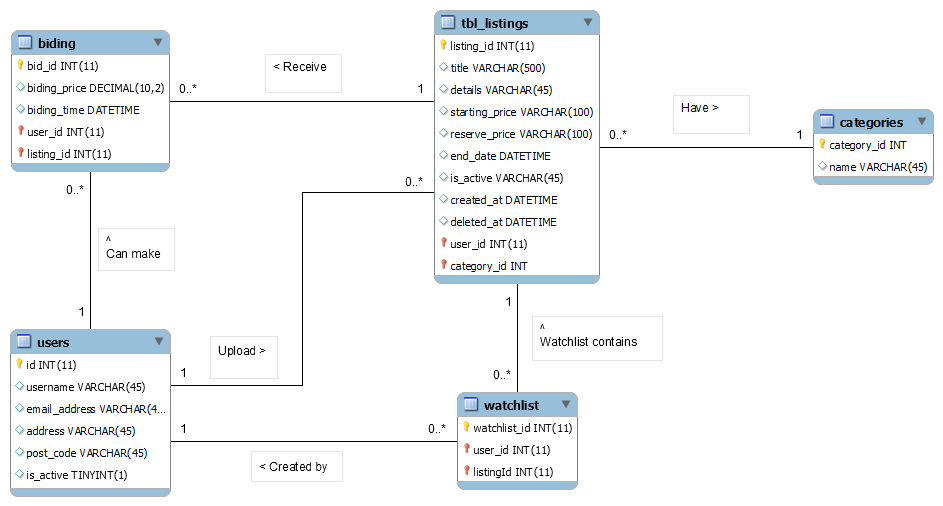
[Normalisation Analysis 5](#_Toc121976434)

[Queries 6](#_Toc121976435)

# Introduction

This goal of this project is to create an auction system website using fundamental database and website creation techniques. This auction system will allow users to perform basic tasks such as registering themselves as a buyer and seller, posting an item for auction and bidding on items. The functioning database system used WAMPServer or XAMPP for the database creation. A partial functional front end was provided, however there was minimal functionality. The website development consisted primarily of PHP, Bootstrap and MySQL for the database back-end development.

# ER Diagram



A few assumptions have been made for the entity relationship diagram about the processes that use the data:

People can register to the auction website as a user who is both a buyer and a seller.

Users are able to post a new listing for an item they want to sell with each listing identified by the key listing\_id.

Every user can bid on listings posted by **other** users provided they have not expired. Ongoing bids can be viewed in the user’s personal bidding dashboard, each bid made is identified by the key bid\_id.

Any user can search for listed items up for auction in a browse listings page.

Users have a ‘recommended’ page where items are recommended to the user based on their bidding history and a watchlist feature which allows users to save a listing they are interested in.

All expired listings are automatically removed from the browse listings page and sellers and buyers are notified of the outcome of the auction; no more bids can be received for that listing.

# Database schema

Once the ER diagram was completed the logical design of the database could be carried out in which a top-down approach was used to translate the conceptual data model into the relational database schema. The name of each entity became the name of the table and each attribute of the entity became a table column. The foreign keys could then be determined by the relationships in the diagram.

|  |  |
| --- | --- |
| users | |
| Key | Description |
| Id {PK} | Primary key that is unique to each user |
| username | The username the user uses to sign up |
| email\_address | Users email address |
| address | Users home address |
| post\_code | Users post code |
| password | Users password used to sign in |
| is\_active | User account active status |

|  |  |
| --- | --- |
| tbl\_listings | |
| Key | Description |
| listing\_id {PK} | Unique id that represents each listing |
| title | Title of the auction |
| details | Details and description of the listing |
| category\_id {FK} | The category of the item being listed |
| starting\_price | The starting price of the listing |
| reserve\_price | The reserve price of the listing |
| end\_date | The date the auction ends |
| user\_id {FK} | Id of the user that lists the item |
| is\_active | The listing is active to view |
| created\_at | The time the listing was created |
| deleted\_at | The time the listing was deleted |

|  |  |
| --- | --- |
| biding | |
| Key | Description |
| bid\_id {PK} | Unique bid id |
| listing\_id {FK} | Which listing was bid on |
| user\_id {FK} | Which user bid on the item |
| biding\_price | The amount of the bid |
| biding\_time | The time the bid was places |

|  |  |
| --- | --- |
| watchlist | |
| Key | Description |
| watchlist\_id {PK} | Unique id associated with each instance of a listing being added to a user’s watchlist |
| user\_id {FK} | The user that watched the item |
| listing\_id {FK} | The listing that is being watched |

|  |  |
| --- | --- |
| categories | |
| Key | Description |
| Category\_id {PK} | Unique id associated with each category |
| name | The name of each category |

# Database relationship table

|  |  |  |
| --- | --- | --- |
| Tables | Relationship | Explanation |
| users -> tbl\_listings | 1 to 0..\* | Each listing can only be created by a single user but each user can create zero or many listings.  Foreign key: user\_id in tbl\_listings. |
| categories -> tbl\_listings | 1 to 0..\* | Each listing must have one (and only one) category\_id, but a category\_id may not be present in any listings or may be present in multiple listings (listings belonging to the same category).  Foreign key: category\_id in tbl\_listings. |
| users -> biding | 1 to 0..\* | Each bid\_id is associated with a single user but each user can make zero or many bids on a listing.  Foreign key: user\_id in biding |
| tbl\_listings -> biding | 1 to 0..\* | Each bid\_id is associated with a single listing but each listing can have zero or many bid\_id’s associated with it.  Foreign key: listing\_id in biding |
| users -> watchlist | 1 to 0..\* | Each watchlist\_id can only be associated with a single user, but a user may have zero or many watchlist\_id’s associated with their account (that each can only correspond to a single listing).  Foreign key: user\_id in watchlist |
| tbl\_listings -> watchlist | 1 to 0..\* | Each watchlist\_id can only be associated with a single listing, but a single listing may have zero or many watchlist\_id’s associated with it (each corresponding to a single user).  Foreign key: listing\_id in watchlist |

# Normalisation Analysis

As can be seen in the tables above, no column of any of the database tables contains multiple values; for a single row and single column, only one piece of data is stored there with each row uniquely identified by a primary key. Therefore the database can be said to be in first normal form. Separate tables have been created for sets of values that apply to multiple records: the users table stores information about users personal information; the tbl\_listings table stores all of the auction information; the biding table stores all bidding information and history; the categories table stores all category names and the watchlist table stores each user’s watchlist history and data. Each of these tables are related to each other using foreign keys; therefore every non-primary key attribute is fully functionally dependent on any primary key and no non-primary key attribute is transitively dependent on any primary key. Hence the database can be said to be in second and third normal form.

# Queries

|  |
| --- |
| Browse.php |
| $query = "SELECT tbl\_listings.listing\_id, tbl\_listings.title, tbl\_listings.details, MAX(biding.biding\_price), tbl\_listings.starting\_price, tbl\_listings.end\_date FROM tbl\_listings LEFT JOIN biding ON tbl\_listings.listing\_id=biding.listing\_id WHERE title IS NOT NULL"; |
| List listing\_id, title, details, starting\_price, end\_date and maximum bidding price for listings which have a non-null title |

|  |
| --- |
| Browse.php |
| $query = "SELECT tbl\_listings.listing\_id, tbl\_listings.title, tbl\_listings.details, MAX(biding.biding\_price), tbl\_listings.starting\_price, tbl\_listings.end\_date FROM tbl\_listings LEFT JOIN biding ON tbl\_listings.listing\_id=biding.listing\_id WHERE title LIKE '%$keyword%'"; |
| List listing\_id, title, details, starting\_price, end\_date and maximum bidding price for listings whose title includes the value of $keyword. |

|  |
| --- |
| Browse.php |
| $query\_result = $query . " GROUP BY tbl\_listings.listing\_id ORDER BY (CASE WHEN (tbl\_listings.end\_date > CURRENT\_TIMESTAMP) THEN TIMEDIFF(tbl\_listings.end\_date,CURRENT\_TIMESTAMP) ELSE ADDTIME((TIMEDIFF(CURRENT\_TIMESTAMP, tbl\_listings.end\_date)),\"10000:0:0\") END) LIMIT $results\_per\_page"; |
| Sort the columns listed in $query by end date of auctions |

|  |
| --- |
| Browse.php |
| $query\_result = $query . " GROUP BY tbl\_listings.listing\_id ORDER BY (CASE WHEN MAX(biding.biding\_price) IS NULL THEN tbl\_listings.starting\_price ELSE MAX(biding.biding\_price) END) LIMIT $results\_per\_page"; |
| Sort the columns listed in $query by latest prices of auctions in ascending order |

|  |
| --- |
| Browse.php |
| $query\_result = $query . " GROUP BY tbl\_listings.listing\_id ORDER BY (CASE WHEN MAX(biding.biding\_price) IS NULL THEN tbl\_listings.starting\_price ELSE MAX(biding.biding\_price) END) DESC LIMIT $results\_per\_page"; |
| Sort the columns listed in $query by latest prices of auctions in descending order |

|  |
| --- |
| Browse.php |
| $temp[1] = "COUNT(DISTINCT tbl\_listings.listing\_id)” |
| To count the number of unique listing\_id values in listing table |

|  |
| --- |
| create\_auction\_result.php |
| $insertQuery = "insert into tbl\_listings(title,details,category\_id,starting\_price,reserve\_price,end\_date,user\_id,is\_active,created\_at) values('$txtTitle','$txtDetails',$ddCategory,'$txtStartPrice','$txtReservePrice','$txtEndDate',$user\_id, 1, '$curdate')"; |
| Insert a new auction with information: $txtTitle','$txtDetails',$ddCategory,'$txtStartPrice','$txtReservePrice','$txtEndDate',$user\_id, 1, '$curdate' |

|  |
| --- |
| Header.php |
| $checkUsername = $mysqli->query("select \* from users where username = '$username' limit 1"); |
|  |

|  |
| --- |
| Listing.php |
| $getListingQuery = "select \* from tbl\_listings where listing\_id = $item\_id"; |
| List the auction whose listing id is equal to the value of $item\_id |

|  |
| --- |
| mybid.php |
| $getUserbidingQuery = "SELECT \* FROM tbl\_listings WHERE EXISTS(SELECT \* FROM biding WHERE tbl\_listings.listing\_id = biding.listing\_id AND user\_id = $user\_id)"; |
| List the auctions the user with $user\_id has bid |

|  |
| --- |
| mybid.php |
| $getBidingNumQuery = "SELECT COUNT(\*) from biding WHERE listing\_id = $listingone"; |
| Count the number of auction whose listing\_id is $listingone |

|  |
| --- |
| mybid.php |
| $$CheckBidQuery = "SELECT \* FROM biding WHERE user\_id = $user\_id and listing\_id = $listingone"; |
| List the biddings on the auction $listingone from user with $user\_id |

|  |
| --- |
| mybid.php |
| $YourMaxBidingQuery = "SELECT MAX(biding\_price) FROM biding WHERE user\_id = $user\_id and listing\_id = $listingone"; |
| List the max biding price which the user with id $user\_id has made for the auction $listingone |

|  |
| --- |
| mylisings.php |
| $getUserListingQuery = "select \* from tbl\_listings where user\_id = $user\_id"; |
| List the auctions created by user with id $user\_id |

|  |
| --- |
| mylisings.php |
| $getBidingNumQuery = "SELECT COUNT(\*) from biding WHERE listing\_id = $listingone"; |
| List the number of biddings on the auction with listing\_id $listingone |

|  |
| --- |
| mywatchlist.php |
| $watchlist\_listing\_query = "SELECT \* FROM tbl\_listings as listings WHERE listings.listing\_id IN (SELECT listing\_id FROM watchlist WHERE user\_id = $id) LIMIT $results\_per\_page"; |
| List the auctions whose listing\_id is added to watchlist by user with id $id |

|  |
| --- |
| mywatchlist.php |
| $num\_watchlist\_query = "SELECT COUNT(listing\_id) FROM watchlist WHERE user\_id = $id"; |
| Count the number of auctions which user with id $id has added |

|  |
| --- |
| mywatchlist.php |
| $count\_bid\_query = "SELECT COUNT(\*) FROM biding WHERE listing\_id = {$row['listing\_id']}"; |
| Count the number of biddings on the auction whose lising\_id is $row['listing\_id'] |

|  |
| --- |
| mywatchlist.php |
| $top\_bid\_query = "SELECT MAX(biding\_price) FROM biding WHERE listing\_id = {$row['listing\_id']}"; |
| Find the highest biding price for the auction whose listing\_id is $row['listing\_id'] |

|  |
| --- |
| place\_bid.php |
| $UpdateNewBidQuery = "UPDATE tbl\_listings SET starting\_price = $biding\_price WHERE listing\_id = $listing\_id"; |
| Update the latest starting price as the highest biding price |

|  |
| --- |
| place\_bid.php |
| $getbiding\_query = "Insert into biding(listing\_id,user\_id,biding\_price,biding\_time) VALUES($listing\_id,$user\_id,$biding\_price,'$biding\_time')"; |
| Create a data row for a new bidding with its information |

|  |
| --- |
| register.php |
| $registerQuery = "INSERT INTO users (username, email\_address, address, post\_code, password, is\_active) VALUES ('$username', '$email\_address', '$address', '$post\_code', '$password', 1)"; |
| Create a data row for new users with its information |

|  |
| --- |
| watchlist\_funcs.php |
| $add\_to\_watch\_query = "INSERT INTO watchlist VALUES ($user\_id, $item\_id[0])"; |
| Add the auction with itemid $item\_id[0] into the watchlist of user with id $user\_id |

|  |
| --- |
| watchlist\_funcs.php |
| $remove\_watch\_query = "DELETE FROM watchlist WHERE user\_id = $user\_id AND listing\_id = $item\_id[0]"; |
| Remove the auction with itemid $item\_id[0] from the watchlist of user with id $user\_id |

|  |
| --- |
| Recommendation.php |
| $recco\_listing\_query = "SELECT \* FROM tbl\_listings WHERE (CURRENT\_TIMESTAMP<end\_date) AND listing\_id NOT IN(SELECT listing\_id FROM biding WHERE user\_id = $user\_id) AND listing\_id IN(SELECT listing\_id FROM biding WHERE user\_id != $user\_id AND user\_id IN(SELECT user\_id FROM biding WHERE listing\_id IN(SELECT listing\_id FROM biding WHERE user\_id = $user\_id))) LIMIT $results\_per\_page"; |
| “you might want to bid on the sorts of things other people, who have also bid on the sorts of things you have previously bid on, are currently bidding on” |

|  |
| --- |
| sendAuctionEnd.php |
| $buyer\_email\_query = "SELECT email\_address, title FROM biding LEFT JOIN tbl\_listings ON biding.listing\_id = tbl\_listings.listing\_id LEFT JOIN users ON biding.user\_id = users.id  WHERE (biding\_price, biding.listing\_id) IN( SELECT MAX(biding\_price), listing\_id FROM biding  WHERE listing\_id IN( SELECT listing\_id FROM tbl\_listings WHERE NOW() > end\_date AND DATE\_ADD(end\_date, INTERVAL 30 MINUTE) >= NOW()) AND starting\_price >= reserve\_price GROUP BY listing\_id)" |
| Select the buyer with the highest bid and the title of the auction when it is finished. The file is run by the task scheduler once per half an hour |

|  |
| --- |
| sendAuctionEnd.php |
| $seller\_email\_query = "SELECT title, email\_address FROM users INNER JOIN tbl\_listings ON users.id = tbl\_listings.user\_id WHERE NOW() > end\_date AND DATE\_ADD(end\_date, INTERVAL 30 MINUTE) >= NOW()"; |
| Select the seller with the finished auction. The file is run by the task scheduler once per half an hour |

|  |
| --- |
| watchlist\_notif.php |
| $watchlist\_email\_query = "SELECT email\_address FROM users WHERE id IN(SELECT user\_id FROM watchlist WHERE listing\_id = $listing\_id)"; |
| Get the emails of people who are watching an item |

|  |
| --- |
| watchlist\_notif.php |
| $buyer\_email\_query = "SELECT email\_address FROM users WHERE id= $user\_id"; |
| Get the emails of people who are watching an item |

|  |
| --- |
| watchlist\_notif.php |
| $listing\_title\_query = "SELECT item\_title FROM listings WHERE listing\_id = $listing\_id"; |
| Gets the title of the auction in question |

|  |
| --- |
| watchlist\_notif.php |
| $outbid\_query = "SELECT email\_address FROM users WHERE id IN(SELECT user\_id FROM bidings WHERE listing\_id = $item\_id AND biding\_price = $previous\_top\_bid)" |
| Gets the email of the outbid buyer |