CSC 4710 – Database Systems

Spring 2021 – XLS Group Y^

Final Project Report - Due April 24th, 2021 11:59 pm

**Team:** Real Estate Database

**Team members**: Subhan Azmi

**Introduction:**

My project is a real estate database that encapsulates the information about transactions between business entities regarding properties. It is meant to be used by real estate professionals as well as anyone else trying to find property specific information.

**Topic:**

The topic for this project is to create a database for real estate properties and their transactions. The main information is of properties and the other major information, transactions, is of sales, leases, and mortgages of properties which involve various parties (buyers/sellers, owners and lenders).

The database contains tables for business entities, properties (including sub types such as commercial, multi\_family, and single\_family) and transactions.

As far as the interface it is partially implemented as well as missing the users/admin login portion as I found that information to be overtly general and did not focus on implementing it into the database:

1. Currently interface allows 3 simple examples of search add and modify.
   1. You can search by property\_id to retrieve its information
   2. You can add/insert a business entity
   3. You can modify/update a business entity via its id.
2. It is primarily a myphpadmin/mysql database with an html access and utilization portal hosted on a local site via Apache web server software.
3. The web pages utilize html to display and allow the user to input data via forms. Most of the html uses graphical and jsquery libraries from the Bootstrap front-end open source toolkit.
4. The backend interaction occurs through AJAX queries in the html files calling upon php files.
5. The php files then connect to the database with preset authentication information and perform the necessary queries and/or return search’s result.

**Scope:**

This database can be used by users of a specific data-service as well as the administrators of the service to look at various data such as transactions or property details. The administrators can utilize the data in the database to have an overview of the market transactions or look at how many transactions between a certain set of dates were sales in which property subtype. The users could be as many as hundreds to more individuals who are looking for specific property or transaction details. It depends on the number of clients the service provider has.

**Implementation:**

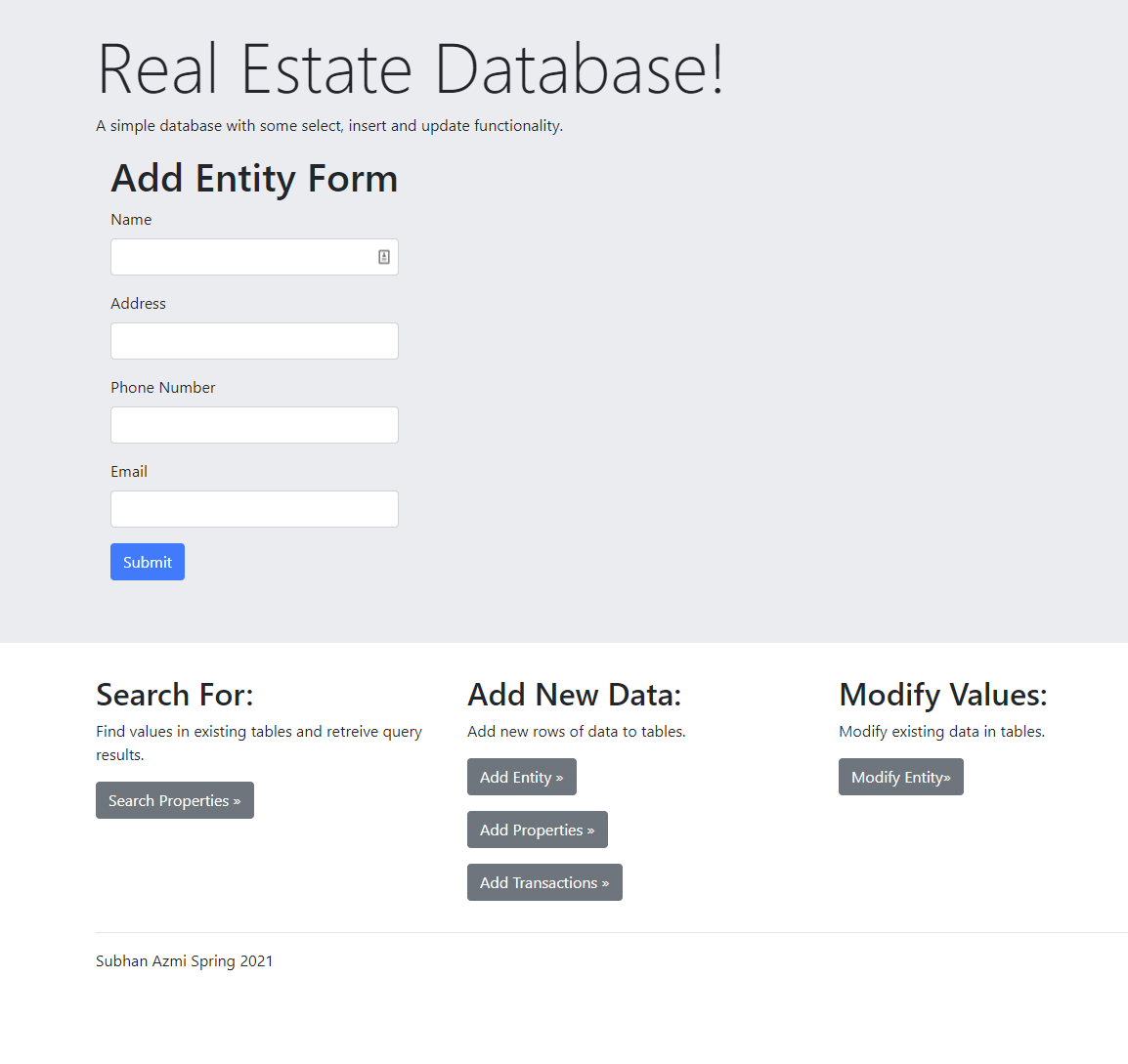
Tables and their attributes:

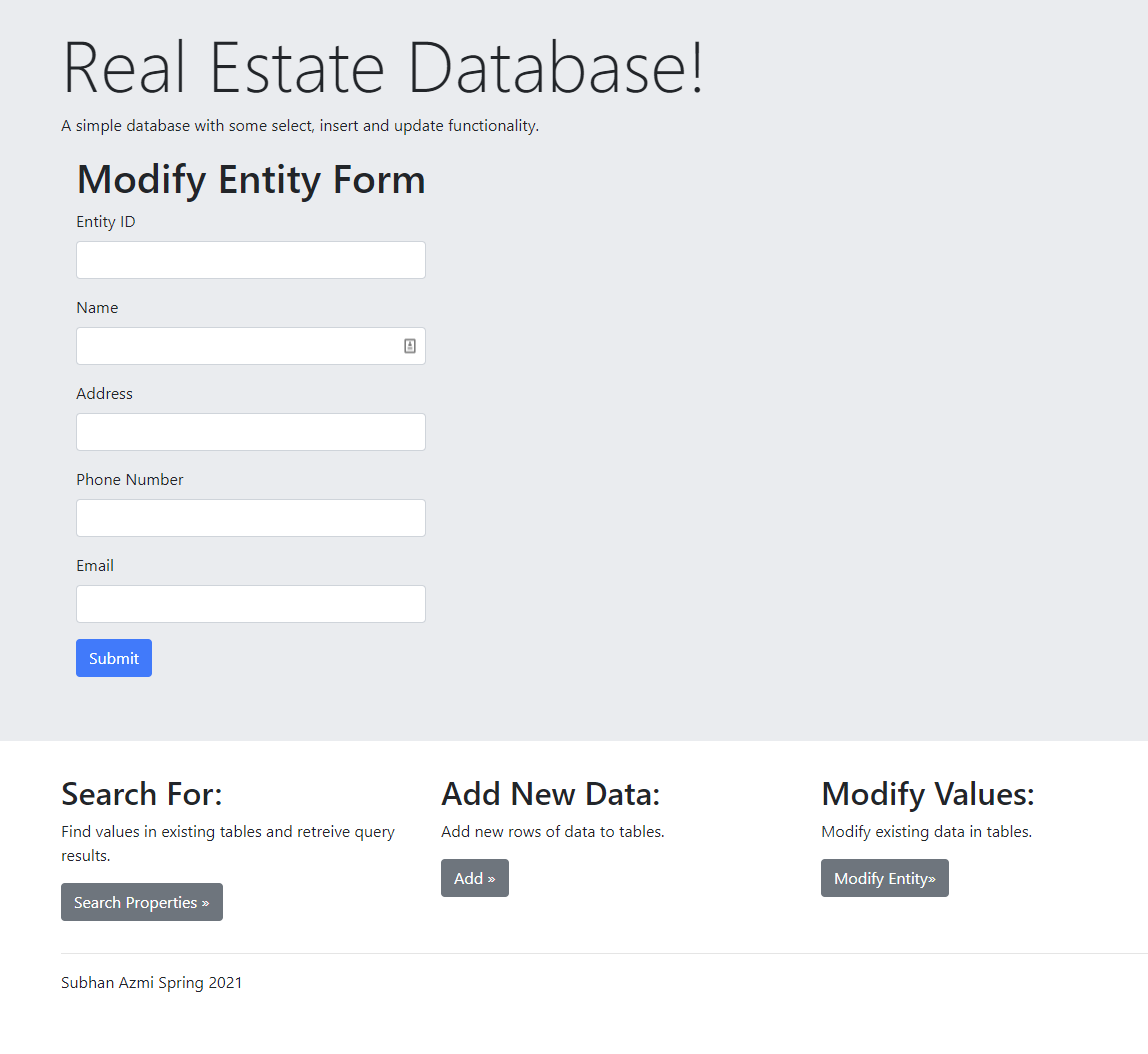
1. Business\_entity:
   1. Be\_id: auto incrementing primary key.
   2. Name: name of entity.
   3. Address: current address of entity.
   4. Phone: current phone of entity.
   5. Email: current email of entity.
2. Owners\_properties\_owned:
   1. Owner\_id: foreign key from Business\_entity table.
   2. Property\_id: foreign key from property table
   3. Start\_date: property was owned by owner starting from this date
   4. End\_date: property was owned by owner until this date.
3. Property:
   1. Property\_id: auto incrementing primary key.
   2. Address: address of property.
   3. Total\_sf: total square foot area of the property.
   4. Total\_acr: total acreage occupied by the property.
   5. Year\_constructed: when the property was constructed.
   6. Last\_sold\_date: when the property was last sold.
   7. Last\_appraisal\_value: last appraisal value of the property.
   8. Floors: how many floors are on the property.
4. Property\_commercial:
   1. Commercial\_id: auto incrementing primary key.
   2. Property\_id: foreign key from property table which can be the same for multiple commercial primary keys as the same building can be sold, renovated and rebranded under a new name. As a subclass inherits the other attributes of property.
   3. Name: name of commercial property.
   4. Total\_units: total units at the property.
5. Property\_multi\_family:
   1. Multi\_family\_id: auto incrementing primary key.
   2. Property\_id: foreign key from property table which can be the same for multiple multi\_family primary keys as the same complex can be sold, renovated and rebranded under a new name. As a subclass inherits the other attributes of property.
   3. Name: name of multi\_family property.
   4. Total\_units: total units at the property.
   5. Style: Either apartments, condos or townhomes.
   6. Avg\_rent: average rent at the complex.
6. Property\_single\_family:
   1. Single\_family\_id: auto incrementing primary key.
   2. Property\_id: foreign key with a 1:1 ratio to primary key.
   3. Style: can be one of many such as “Victorian” or “Colonial”.
   4. Bedrooms: how many bedrooms are at the property.
   5. Bathrooms: how many bathrooms are at the property.
7. Transactions\_sale: (All transactions can only be of one type.)
   1. Transaction\_sale\_id: auto incrementing primary key.
   2. Property\_id: foreign key indicating which property this transaction relates to.
   3. Seller\_id: business\_entity foreign key indicating which entity sold the property.
   4. Buyer\_id: business\_entity foreign key indicating which entity bought the property.
   5. Sale\_price: the price at which the transaction is completed.
   6. Sale\_date: the date on which the transaction is completed.
8. Transactions\_mortgage: (All transactions can only be of one type.)
   1. Transaction\_morgage\_id: auto incrementing primary key.
   2. Property\_id: foreign key indicating which property this transaction relates to.
   3. Lender: business\_entity foreign key indicating which entity holds the deed of the property.
   4. New\_owner: business\_entity foreign key indicating which entity will own the property after paying it off.
   5. Start\_date: start of mortgage time period.
   6. End\_date: end of mortgage time period.
   7. Time\_period\_months: length of the mortgage.
   8. Down\_payment: amount put forth at beginning of mortgage towards principal.
   9. Amount\_mortgaged: amount of overall principal which is loaned to complete the purchase and must be paid off.
   10. Interest\_percent: interest rate of the mortgage.
9. Transactions\_lease:
   1. Transaction\_lease\_id: auto incrementing primary key.
   2. Property\_id: foreign key indicating which property this transaction relates to.
   3. Owner\_id: business\_entity foreign key indicating which entity owns the property.
   4. Leasee\_id: business\_entity foreign key indicating which entity is leasing the property.
   5. Sf\_leased: square foot that is leased by the leasee.
   6. Unit: Which subunit of the property does this transaction refer to.
   7. Start\_date: start of lease time period.
   8. End\_date: end of lease time period.
   9. Time\_period\_months: length of the lease.
   10. Price\_per\_month: amount the leasee must pay to the owner every month.

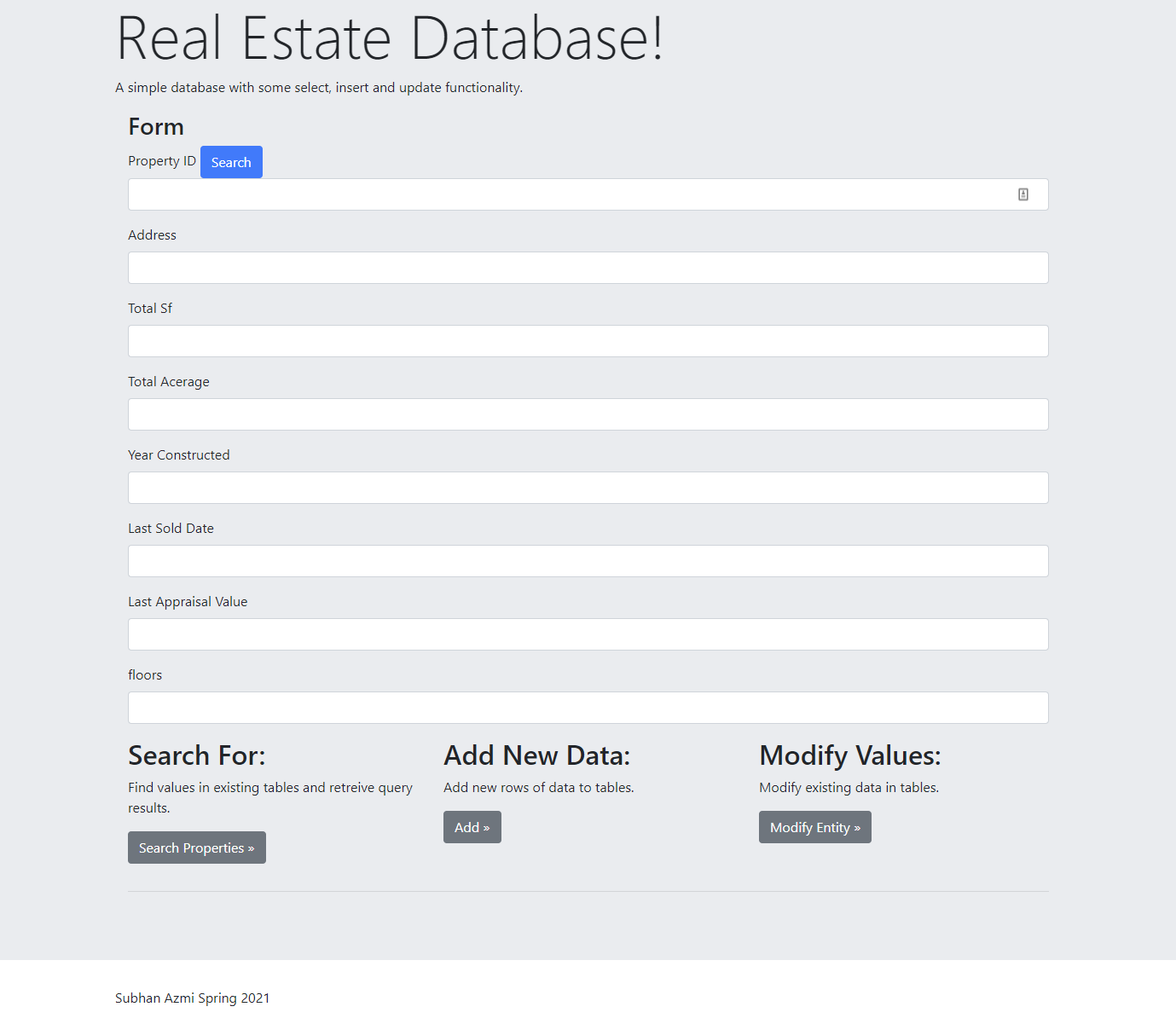
Interface implementation:

1. The first thing needed was installation of the php language which is simply downloading the language reference files and ensuring the path variable in windows is set correctly to interpret all php commands via the reference files.
2. I setup Apache web server by downloading the necessary files and unzipping them in the right location. I also changed some configuration files in the conf folder to ensure that php files are correctly read and utilized.
3. Phpmyadmin is the crux of the entire implementation. It is hosted on the apache web server and is a direct interaction with the MySql installation on windows.
   1. I simply had to ensure that the unzipped files were placed in the correct apache folder on the machine to run.
   2. I used this to create the tables and connect the foreign keys.
4. The actual interface is built off of html web pages and their supporting php files.
   1. They reference many of the bootstrap libraries and need to be placed in a specific hierarchal way to run correctly.
   2. The files must be in a directory which is parallel to the bootstrap folder.
   3. Main.html
      1. Displays the main page of the interface.
   4. Searchpageproperty.html
      1. Allows user to input an id into a form and then with a click of the button runs the searchproperty.php to retrieve information for said and display it in the text boxes.
      2. The ajax jsquery posts the value of the id text box to the searchproperty.php and processes the returned parameter/data.
         1. This then assigns the appropriate portions of the returned data to the visible textboxes to display the results.
   5. Modifypageentity.html
      1. Allows user to input id, name, address, phone and email into text boxes and then with the push of a button runs the modifyentity.php.
      2. The ajax jsquery posts the values of the form to modifyentity.php. Displays an “update was submitted” on successful operation.
   6. Addpage.html
      1. Incomplete but allows user to choose between add\_entity, add\_property, and add\_transaction.
      2. Currently only works for add\_page\_entity.
   7. Addpageentity.html
      1. Allows user to input name, address, phone and email into text boxes and then with the push of a button runs the addentity.php.
      2. The ajax jsquery posts the values of the form to addentity.php. Displays a “form was submitted” on successful operation.
   8. Addentity.php
      1. This php file connects to the database with pre-set login credentials and processes the insert query with the passed name,address,phone, and email parameter.
      2. Creates a new record in the business\_entity table.
      3. INSERT into business\_entity(name, address, phone, email) values(?, ?, ?, ?)
   9. Modifyentity.php
      1. This php file connects to the database with pre-set login credentials and processes the update query with the passed id,name,address,phone, and email parameter.
      2. If the id exists then it will update that record.
      3. UPDATE business\_entity SET name='$name', address='$address', phone='$phone', email='$email' WHERE be\_id='$e\_id'
   10. Searchproperty.php
       1. This php file connects to the database with pre-set login credentials and processes the search query with the passed id parameter.
       2. If the id exists and the query is successful then the result’s parameters are assigned to another array which is then returned as a json\_encode.
       3. SELECT \* FROM property WHERE property\_id='$id'

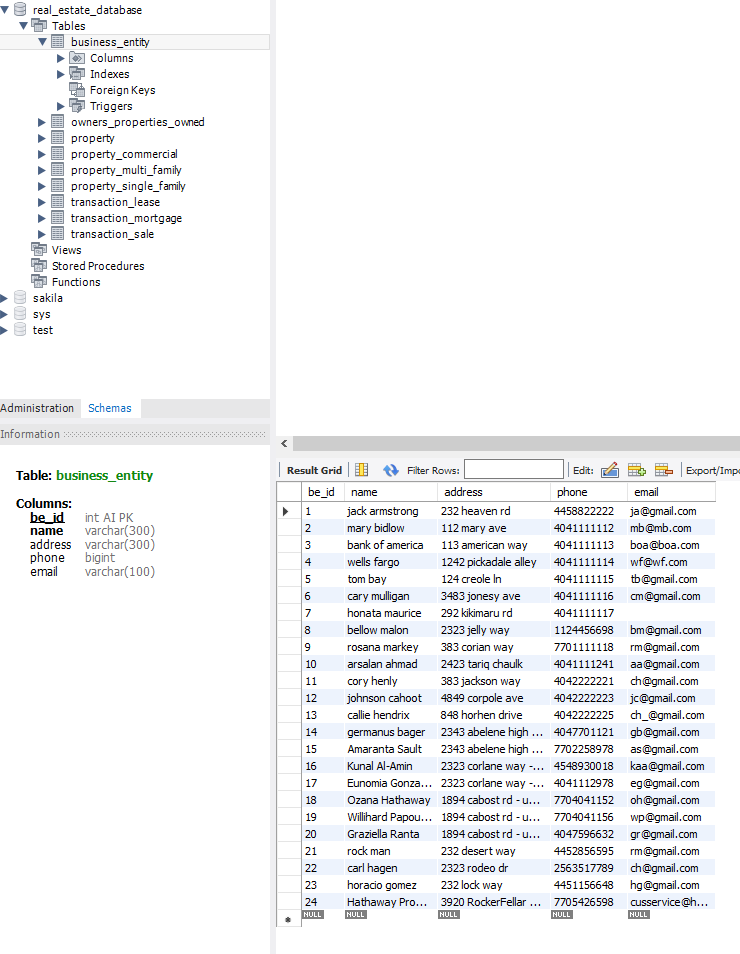
Interface screenshots:



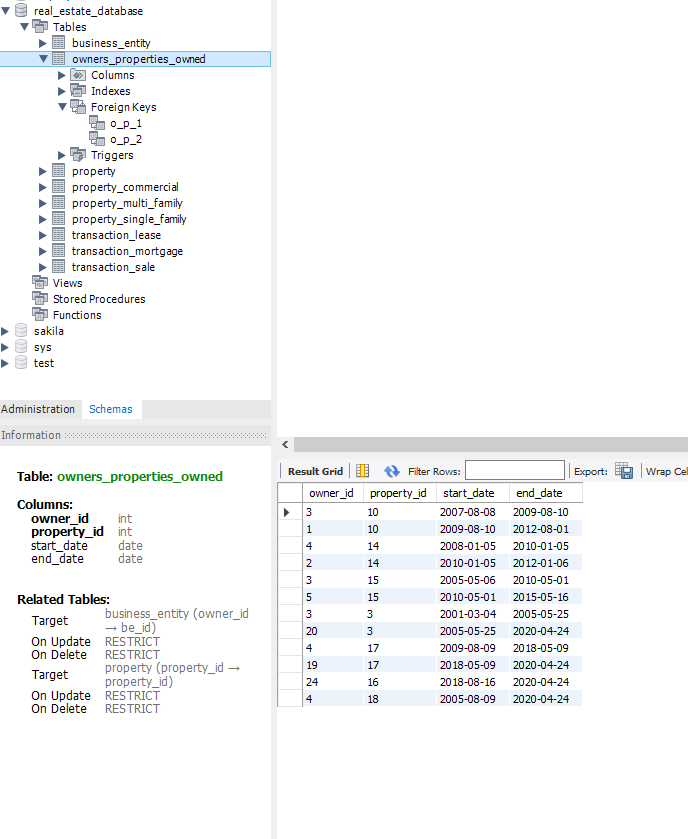




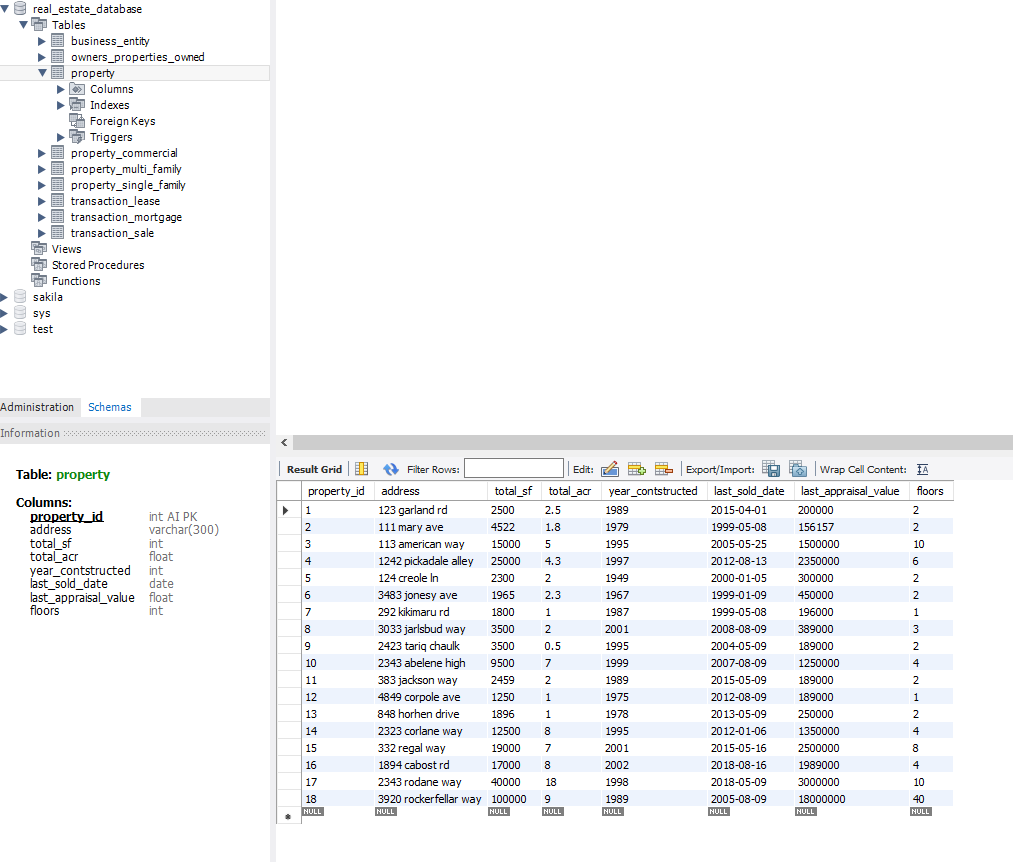
Database: MySql workbench screenshots showcasing foreign keys and tuples

Business\_entity: 

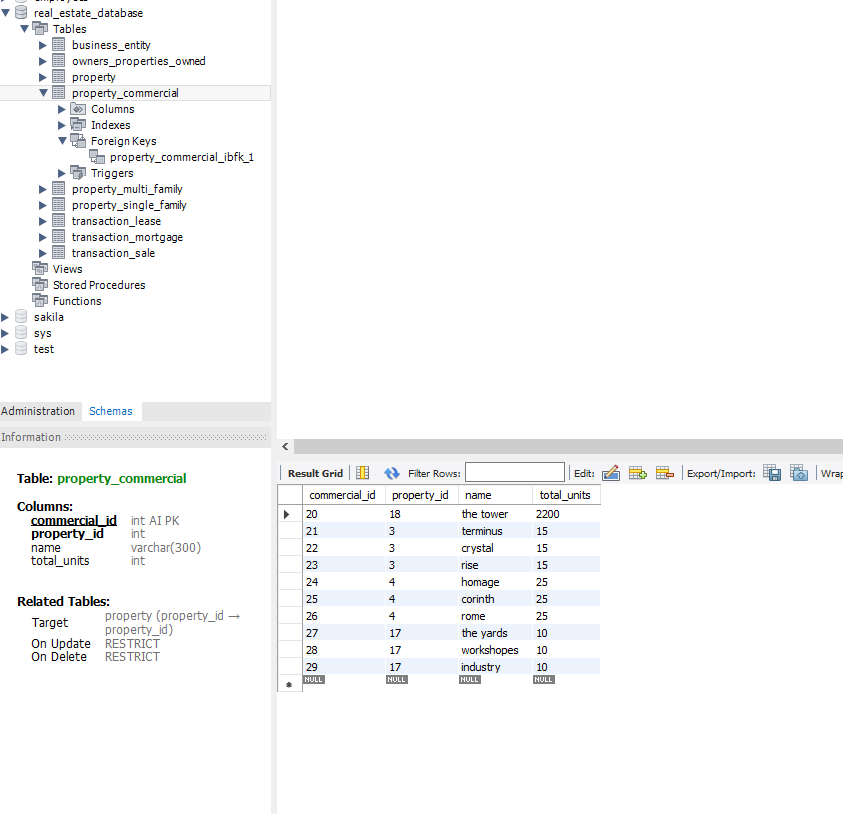
Owners\_properties\_owned:



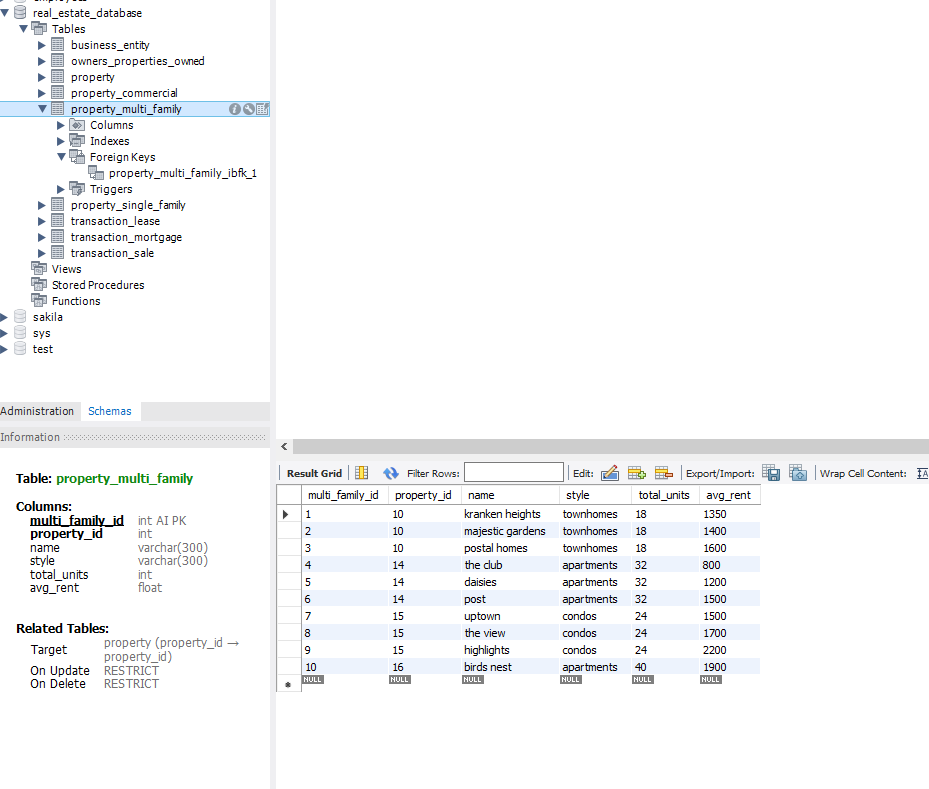
Property:



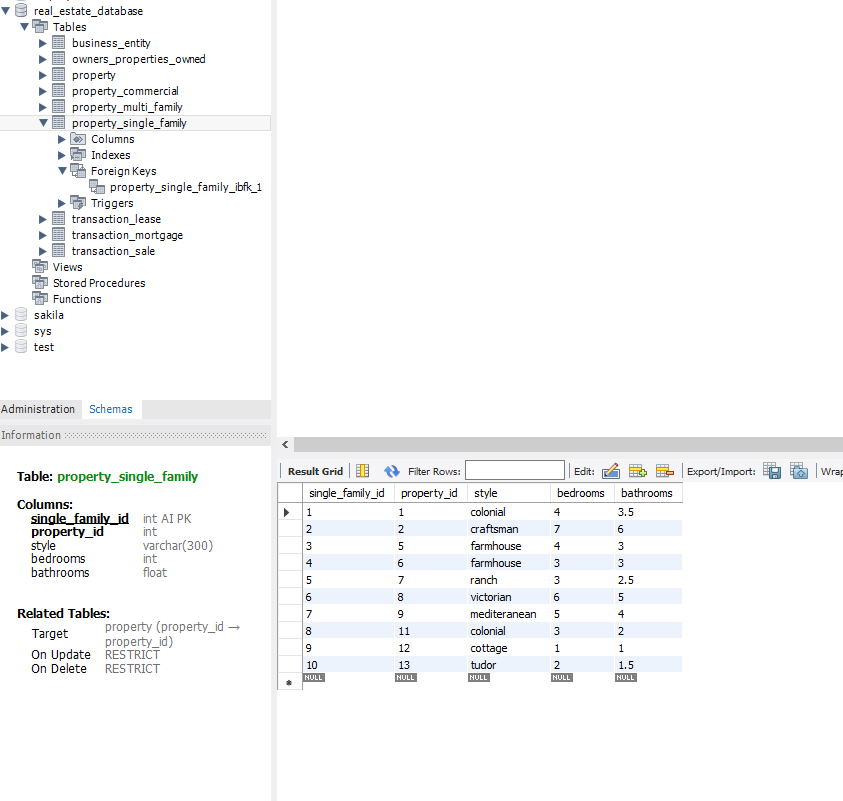
Property\_commercial:



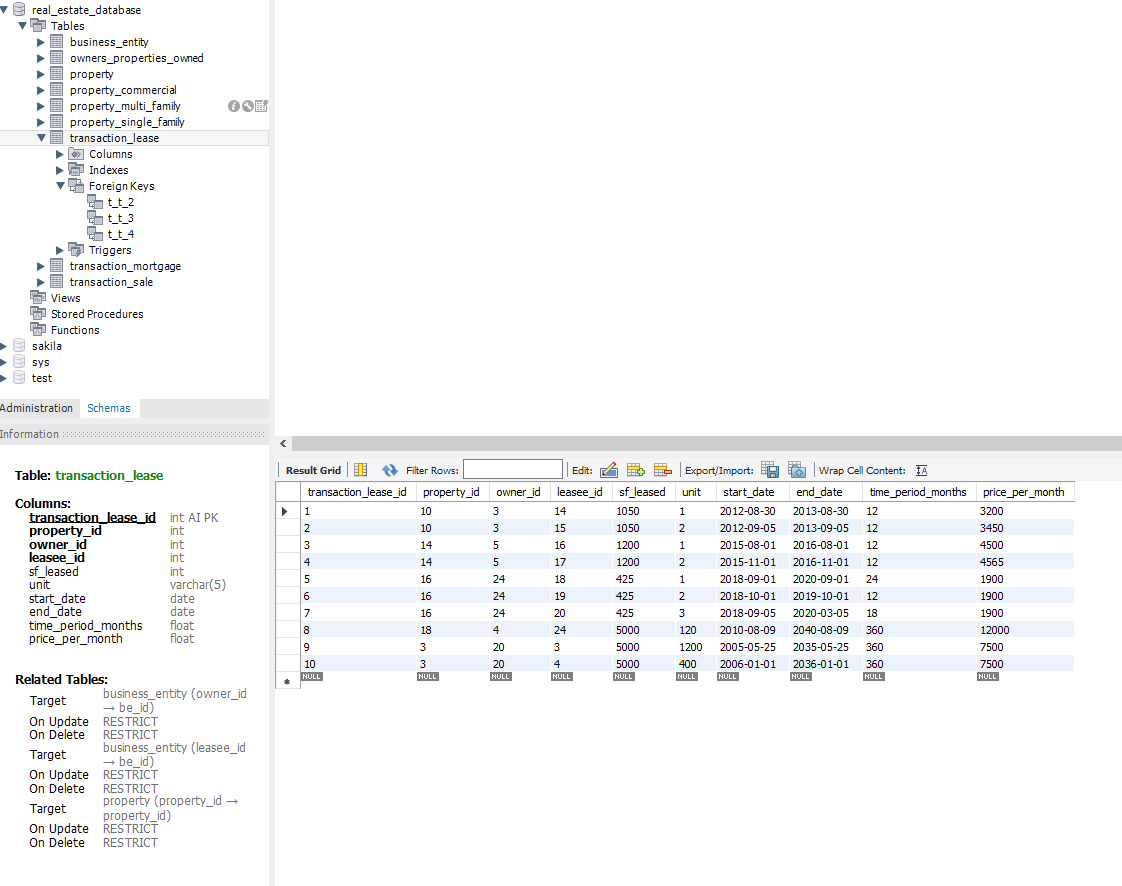
Property\_multi\_family:



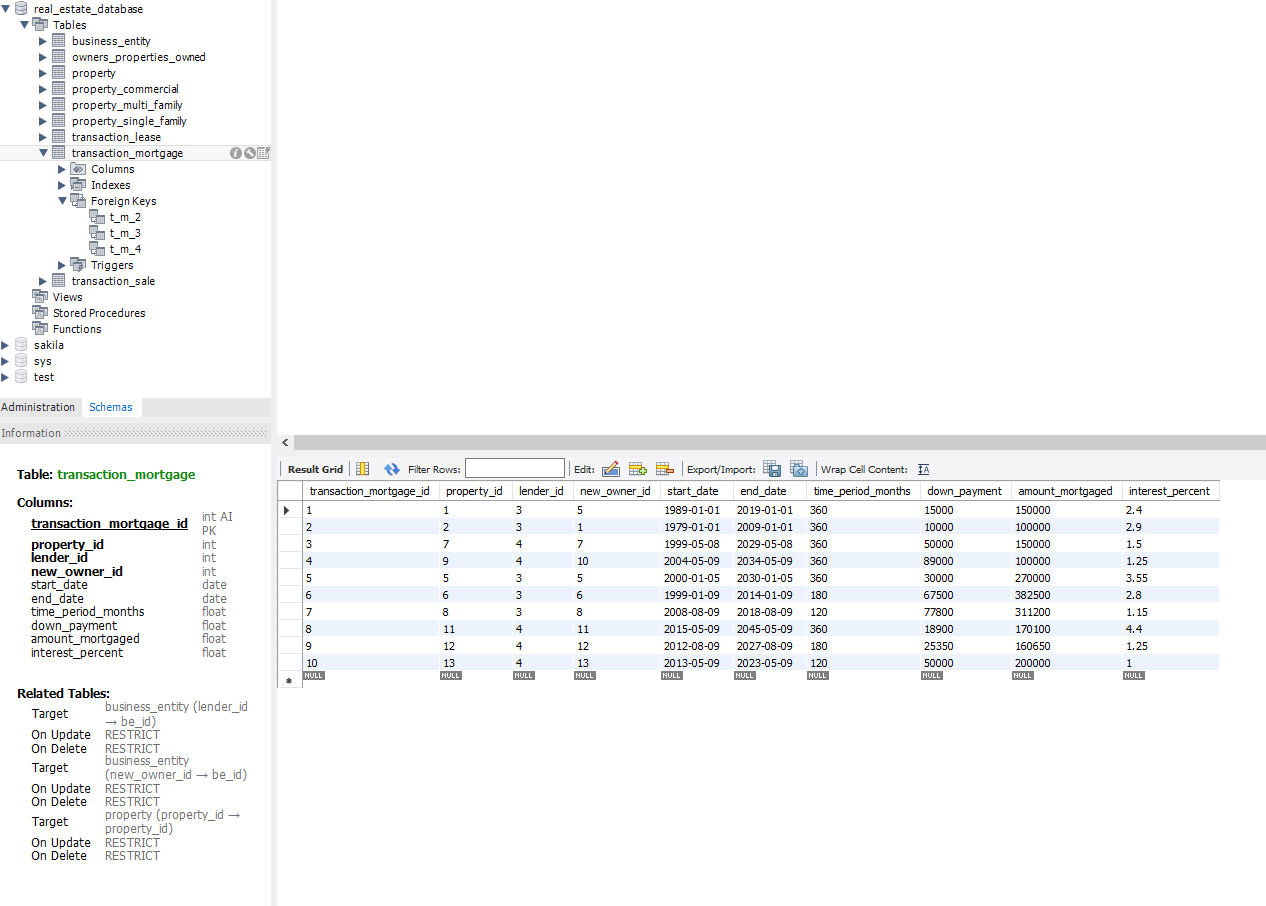
Property\_single\_family:



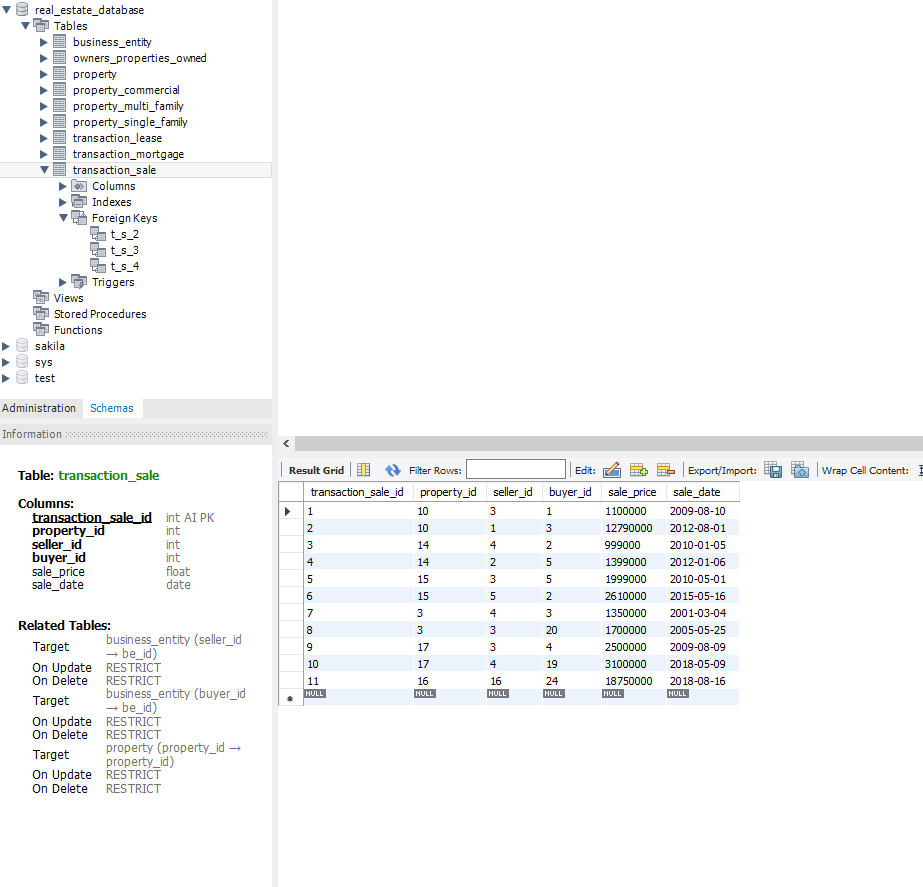
Transaction\_lease:



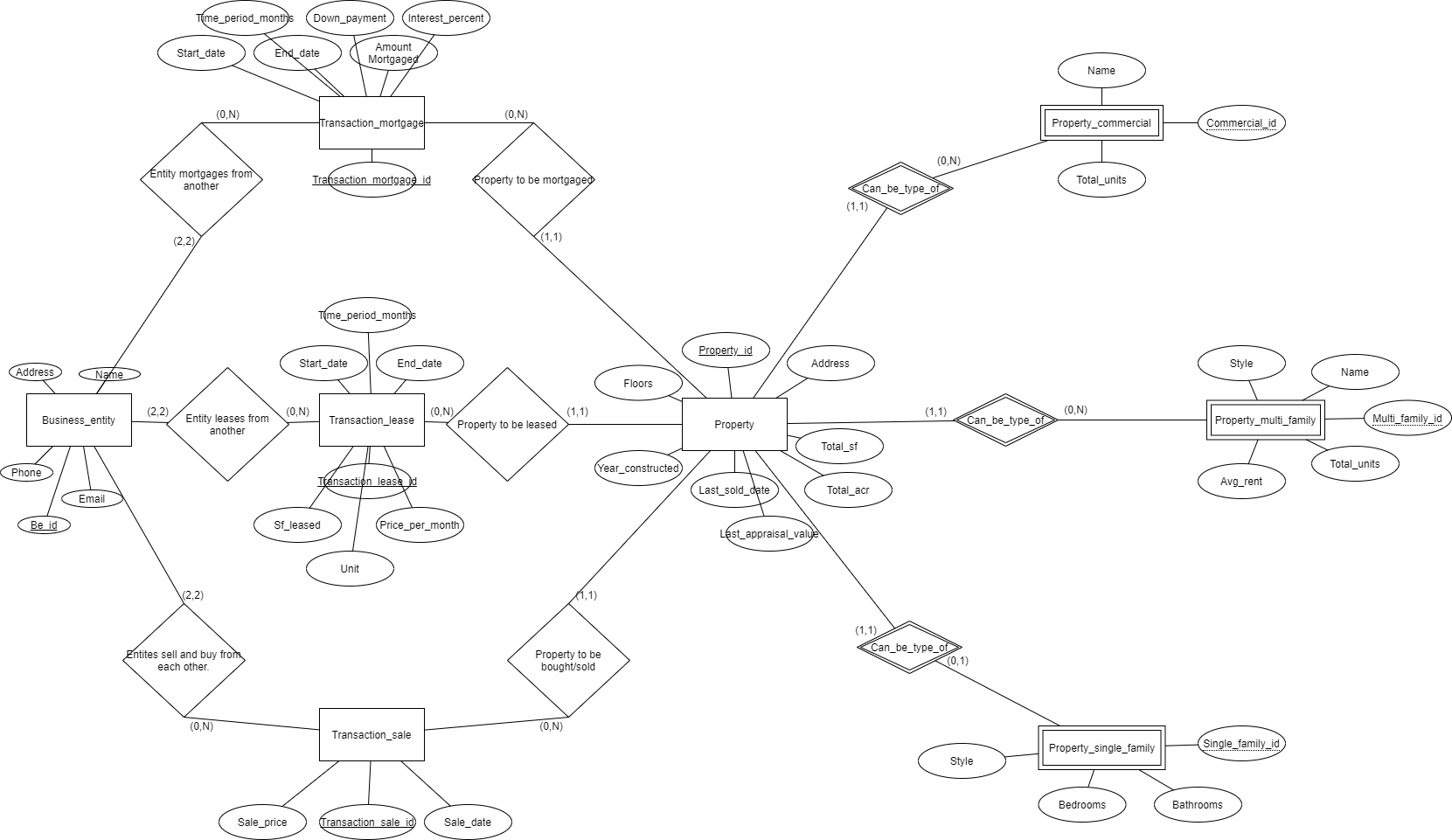
Transaction\_mortgage:



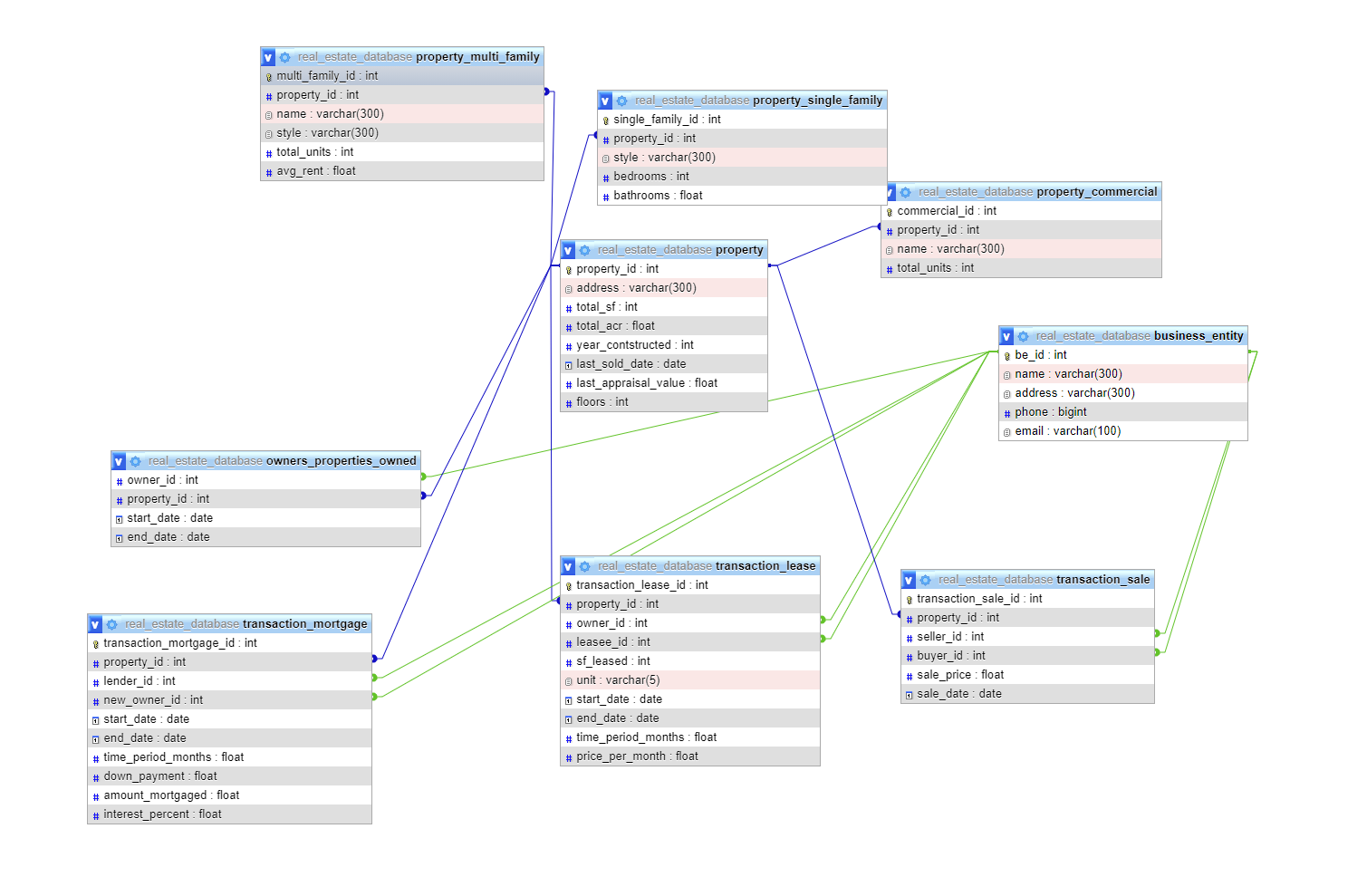
Transaction\_sale:



**ER diagram:**



**Phpmyadmin designer view showcasing foreign keys:**



**Use cases:**

1. The most important feature set of this database is tracking transactions. This can provide information as to which parties were involved in a given transaction and the transaction details.
2. Property details can be searched for using id or even address.
3. This database can be used to keep track of owners of properties, or to track who owned which property from when till when.
4. An important task of multi-family real estate is tracking current leases. A view can be created using the transaction\_lease table to showcase who is currently leasing at a specific property. By matching address in property table to retrieve property id as part of a sub query one can then use that id to see which leases are currently active by filtering leases only at that given property id.

**Conclusion:**

In conclusion this database is fairly useful and tactile for people who want to store and find information about properties and transactions and the parties involved. There are obvious areas of improvement such as the interface still needs appropriate implementation to allow proper searches, adds, and updates. Another vital way to improve the database would be to separate the data by states or cities and house data related to said states in a central database and in that location as well.

**References:**

Book:

Elmasri, R., & Navathe, S. (2015). *Fundamentals of Database Systems* (7th ed.). Pearson.

How to setup apache youtube video:

*How to Install Apache Server on Windows*. (2015, July 28). [Video]. YouTube. https://www.youtube.com/watch?v=LL2HXgxk6-Q&t=1s

How to setup myphpadmin youtube video:

*How to Install phpMyAdmin on Windows*. (2016, August 21). [Video]. YouTube. https://www.youtube.com/watch?v=hqfIksHKPPg

Youtube video showing how to write php code:

*How to Connect HTML Form with MySQL Database using PHP*. (2019, March 31). [Video]. YouTube. https://www.youtube.com/watch?v=2HVKizgcfjo

Website for search implementation:

S. (2018, January 1). *How to Create a Data Search and Display it to Textbox with jQuery AJAX*. How to Create a Data Search and Display It to Textbox with JQuery AJAX. https://steemit.com/utopian-io/@simpleawesome/how-to-create-a-data-search-and-display-it-to-textbox-with-jquery-ajax

W3schools add to database via php:

*PHP MySQL Insert Data*. (2021). PHP MySQL Insert Data. https://www.w3schools.com/php/php\_mysql\_insert.asp

*PHP MySQL Update Data*. (2021). PHP MySQL Update Data. <https://www.w3schools.com/php/php_mysql_update.asp>

Various softwares:

Apache: https://httpd.apache.org/

Phpmyadmin: <https://www.phpmyadmin.net/>

Bootstrap: <https://getbootstrap.com/>

Php: https://www.php.net/