

Database and Application for Apartment Rent and Lease

Team9: Da chen, Yijun Lin, Yang Yang

2020 Fall, Kansas State University

1 Introduction

An application for apartment rent and lease is developed based on an apartment database system. There are three main functions: 1. The clients can search the apartments to rent based on interested apartment properties. 2. The clients can also post their apartment information for lease in this application. 3. we use the report query to provide information inferred from the data to help our clients make better decisions. Besides, we have a login system for tenants who decide to live in the apartments recorded in the database. We also built a review system so that tenants can rate in a score out of 5 to show whether they're satisfied with their current apartments.

The potential users are targeted based on three aspects. For the leasing function, we target the apartment owners (e.g. property management companies). The rent function is open to customers with the need to rent an apartment, generally, anyone who visits the interface. The review system is only available to tenants.

2 Technical Description

The database is built and maintained by using Microsoft SQL Server. The user interface is implemented in Visual Studio 2019 using C#. The data is simulated in C# as well.

3 Database Design

The database design is shown in Figure 1 below. The database consists of 10 tables and the Apartment table is the main relation with more than 1000 instances. The relationship is shown by the crow's foot notation. Most relationships between the relations are one-to-many and we have three pairs of many-to-many relationships which are connected by three bridge tables, i.e. ResideRecords, ApartmentFeatures, BuildingFeatures. All features are defined with NOT NULL. All primary key, unique key, and foreign key constraints, as well as column data type, are marked in the figure. The sizes of the relations are shown in the data generation section.

All relations support SELECT and INSERT operations except Cities table, FeatureA, and FeatureB tables which are static and only allow to use SELECT. The Cities table contains 10 cities in Kansas. FeatureA and FeatureB tables which contain available amenity information for apartment or building, are not accessible to users to change. There are three instances in FeatureA and the feature names are 'WithBalcony', 'WithWasherDryer', and 'PetFriendly', which are features of apartments. There are two instances in FeatureB and the feature names

are 'WithGym' and 'WithPool' which are features of buildings. The ResideRecords table supports Update to allow customers to give review scores (out of 5) to their current apartments during their check-out. The Apartment table also supports update operations because the reviews given by current residents can change the average review scores of the apartments.

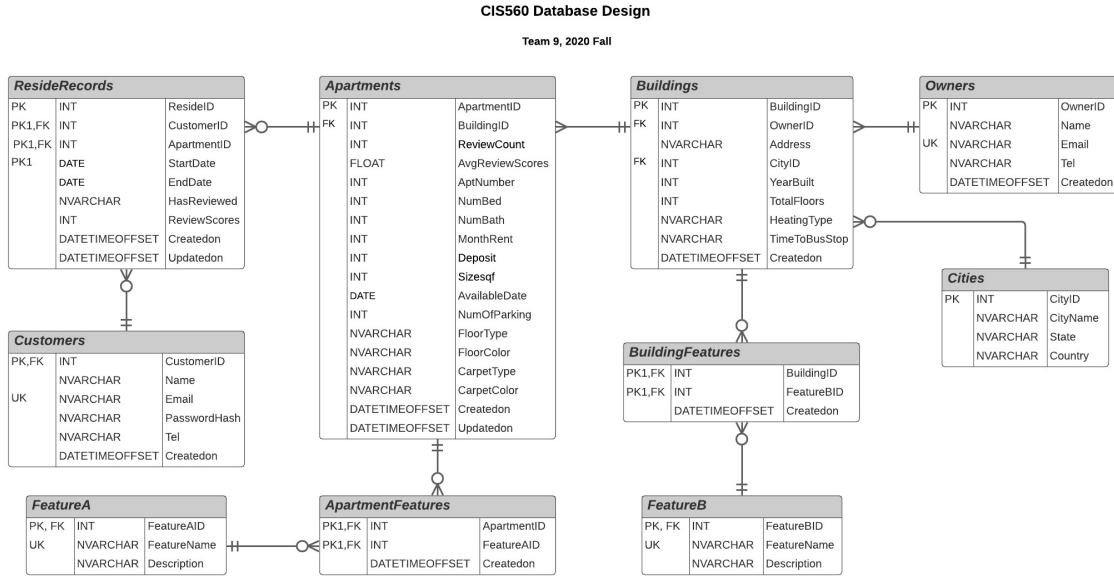


Figure 1: Database Design

4 Data Generation

Since we didn't find a database that fits our database design perfectly, we decided to generate our own data. We use C# in Visual studio to print data that can be used to populate the database.

For the convenience of coding, the data is closer but not exactly as the data in real life. We choose 10 cities in Kansas, which are Kansas City, Topeka, Lawrence, Olathe, Manhattan, Salina, Dodge City, Garden City, Haysville, and Emporia. We assume there are 10 property management companies in each city. Every management company owns 2 buildings with 5 apartments in each building. We fabricate personal information for 100 residents and assume each resident has 2 records of residing in apartments stored in our database. In total, we have 10 instances in Cities table, 100 instances in Owners table, 200 instances in Buildings table, 1000 instances in Apartments table, 100 instances in Customers table, and 200 instances in ResideRecords table.

To generate data which is closer to reality, we use linear models in statistics to generate values for apartment size and month rent. We assume that apartment size is in positive relationships with the number of bedroom and bathroom, and month rent is positively related not only to size but also the year of the building. we use random error to represent other factors that may affect apartment size and month rent.

The linear models are shown as,

$$\mathbf{y}_1 = 860 + 215\mathbf{x}_1 + 108\mathbf{x}_1 + \varepsilon_1 \quad (1)$$

where $\mathbf{y}_1 = (y_1, \dots, y_{1000})^T$ is a 1000×1 vector of apartment size for the population of 1000 instances in Apartment table. The value of \mathbf{y}_1 depends on a base line 860, covariate \mathbf{x}_1 which

is a vector of number of bedroom, covariate x_2 which is a vector of number of bathroom, and a random error ε_1 which follows Gaussian distributions with parameters, mean of 0 and standard deviation of 65.

$$y_2 = -19610 + 100x_1 + 80x_2 + 10x_3 + \varepsilon_2 \quad (2)$$

where $y_2 = (y_1, \dots, y_{1000})^T$ is a 1000×1 vector of month rent in Apartment table. The value of y_2 is calculated from a base value -19610 , covariate x_1 which is a vector of number of bedroom, covariate x_2 which is a vector of number of bathroom, covariate x_3 which is a vector of the year of built, and a random error ε_2 which follows Gaussian distributions with parameters, mean of 0 and standard deviation of 100.

The coefficients for covariates and parameters for ε in the linear models are assigned after a few attempts to generate reasonable ranges of y values. This random error ε is to count for factors which are not in the model and also ensure a variety for y values.

The advantages of the model-based data generation can be seen in the first report query where the month rent is increasing as the room numbers increase, which mimics the real-world data.

5 System Design

The interface is implemented using window form in C#. To operate on the interface, we use buttons to handle click events and switch between different interfaces. The TextBox is employed for input from users. The insert or search functions are implemented by connection to SQL code. The searched information or report query results are displayed as tables in DataGridView. All data are obtained from SQL database or interface, for example, obtain user input via SqlDataReader. We use the library named System.Data.SqlClient to connect to SQL database. The SQL codes are stored in form of string in C#. The results returned from SQL queries are stored in the database in C# and shown in DataGridView.

A system design diagram (Figure 2) illustrates the components in the application and the ways to process tasks and data. The application components are organized and connected by the arrow lines. We will show some example scenarios to further explain the system in the System Features and Usage section.

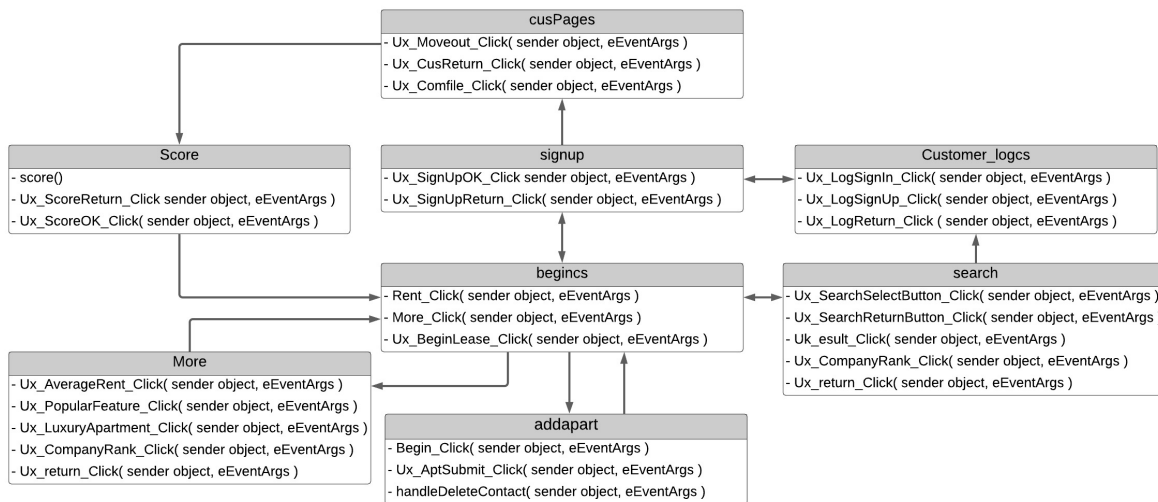


Figure 2: System Design

6 System Features and Usage

A tour to explore the application interface will be presented in this section.

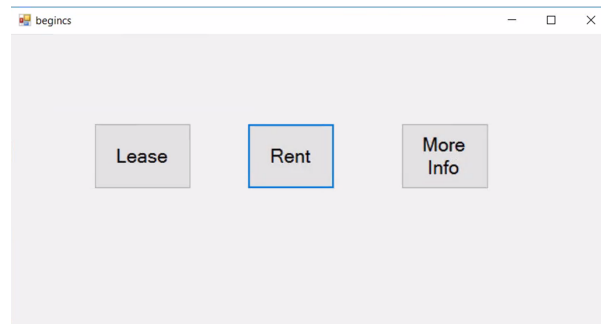


Figure 3: Main Page

The main page of the application is shown in figure 3. There are basically three functions represented by these three buttons. The Lease button is for the clients who want to release their apartment information for leasing. The Rent button is for people with the need to find a apartment to live in. The MoreInfo button, short for More Information, is to show the report queries and can be used as references for clients and other users(e.g. for survey purposes).

A screenshot of a web application window titled "addapart". The form contains the following fields and options:

- Name: zoo
- Email: zoo@ksu.edu
- Tel: 86453112
- City: Haysville (dropdown)
- BuiltYear: 2015
- TotalFloors: 5
- Street: 1836 PL
- Apt Num: 12
- NumBed: 3 (dropdown)
- NumBath: 2 (dropdown)
- MonthRent: 1250
- Deposit: 1500
- SizeSqFt: 1500
- AvailTime: 2020-12-12 (YYYY-MM-DD)
- FloorType: Ceramic (dropdown)
- FloorColor: Brown (dropdown)
- CarpetType: Olefin (dropdown)
- CarpetColor: Warm (dropdown)
- HeatingType: Boiler (dropdown)
- NumOfParking: 1 (dropdown)
- TimeToBusStop: less than 5 min. (dropdown)

At the bottom right, there are two buttons: "Submit" and "Return".

Figure 4: Lease Form

Once click the Lease button in main page, the Lease Form page shows up as Figure 4. In this page, the clients can fill out the form by inputting the apartment information and their contact information as well. The Submit button will insert the input information into our database. The Return button will return back to the main page.

Figure 5 is the search page after clicking on the Rent button in the main page. The users can fill the filtering features based on their preference and every filter can leave to be blank if there is no requirement on it. The Search button will execute the SQL question query and return the result below. If the user finds a satisfying apartment, he can click sign in for further operations. The Return button returns to the main page.

ApartmentID	Address	City	NumOfBedroom	NumOfBathroom	TimeToBusStop	Available
209	3502MACDOUGAL_STREET	Haysville	2	1	10 - 15 min.	9/3/20
216	3504WATTS_STREET	Haysville	2	1	more than 15 min.	12/9/20
256	3512PAGE_STREET	Haysville	2	1	10 - 15 min.	3/9/20

Figure 5: Search Page

Figure 6: Invalid Sign In

Figure 7: Sign Up

The sign in page is used for users to login with email and password. The invalid login error handling is supported and shown in Figure 6. For new users, Sign Up button will lead to a sign up page which requires to fill in the personal information as in Figure 7.

After login, the user can find more detailed information about a certain apartment by input the ApartmentID found in search page and click Check button (Figure 8). If the user is not living in apartments in our database, then the Confirm button is active and the Move out button is inactive (Figure 8). If the user decides to choose this apartment to live in, he can click Confirm to check in this apartment. If the user is a current tenant, the Move out button is now active and confirm is inactive because we don't support duplicated residence (Figure 9). If the tenant decides to move out, he can click the Move out button and will go to the move out and

Figure 8: New User and Check in

Figure 9: Resident and Move out

review page (Figure 10).

Figure 10: Review System

For the convenience of implementation, the review system page is accessible to residents only when they want to move out. They can rate the current apartments in scores out of 5 (Figure 10). The new review score given by a tenant will change the average review score of this apartment in the Apartment table. Since we track the count of review scores for this apartment in the past using ReviewCount feature in Apartment table, the new average review score in Apartment table can be calculated as

$$\frac{ReviewCount \times AverageReviewScores + NewReviewScore}{ReviewCount + 1} \quad (3)$$

if we denote the inserted ReviewScores in ResideRecords table as NewReviewScore. Then the ReviewCount needs to be increased by 1.

Figure 11: Report Query

The More Info button in the main page links to the report query page as shown in Figure 11. There are four buttons for four report queries. The result will be displayed to the right after clicking the according button. The description and analysis of the report queries will be discussed in the report queries section.

There are several strengths of the system. The interface design is concise making it easy to understand and follow. There are several pages for users to explore which are connected logically. The functions of the system covered most of the needs of users. The system is applicable in real life.

The limitation of the system is that we have to memorize the searched ApartmentID during login in order to type in the check information page. It can be improved by finding a method that stores the selected apartment information, and after login, the information can be passed to the confirm page automatically. Another part that needs to be improved is the Return button, Return button is used to return to main page in all pages except one in the sign up page which returns to sign in page. An improvement is to set a Return to Main Page button and a Return to Previous Page button. Due to the time limitation, we haven't done this, but it can be a good way to improve the interface design.

7 Report Queries

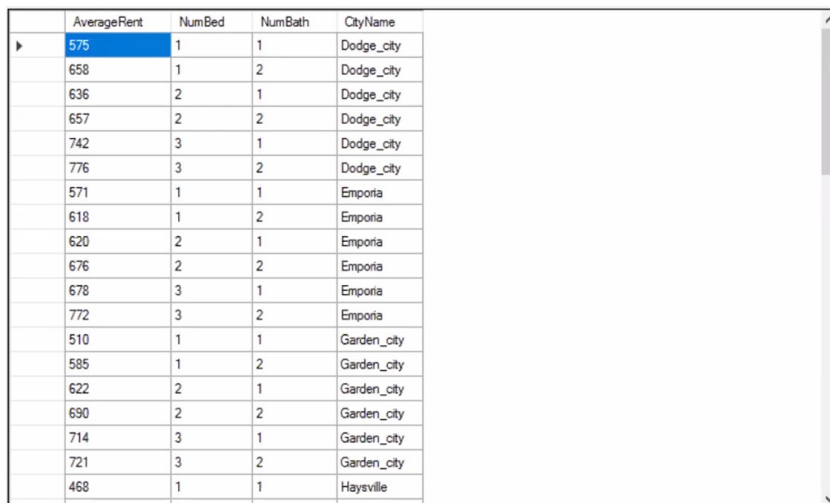
The report queries are to extract valuable information from data in the apartment database system as a reference not only for clients but also for industry and survey use.

7.1 Average Rent

This query shows the average month rent of all different apartment unit types in different cities. The users could use this information as a reference when renting or leasing.

All apartments will be grouped by unit types and cities to calculate the average month rent by inner joining Cities, Buildings, and Apartments tables.

The output is shown in the Figure 12. The average month rent is listed by order of cities then unit types (i.e. number of bedrooms and number of bathrooms). For instance, if a customer wants to find an apartment of 1 bedroom and 1 bathroom in Dodge City, then this report query gives the average month rent of \$ 575 as reference.



AverageRent	NumBed	NumBath	CityName
575	1	1	Dodge_city
658	1	2	Dodge_city
636	2	1	Dodge_city
657	2	2	Dodge_city
742	3	1	Dodge_city
776	3	2	Dodge_city
571	1	1	Emporia
618	1	2	Emporia
620	2	1	Emporia
676	2	2	Emporia
678	3	1	Emporia
772	3	2	Emporia
510	1	1	Garden_city
585	1	2	Garden_city
622	2	1	Garden_city
690	2	2	Garden_city
714	3	1	Garden_city
721	3	2	Garden_city
468	1	1	Haysville

Figure 12: Average Rent

7.2 Popular Feature

The query results in the most frequently equipped properties in the apartments with review scores more than the average score, which we think of this as a popular apartment equipment. This query is useful to the industry for better product design.

In total, we have 5 different feature types, i.e. heating type, floor type, floor color, carpet type, carpet color. In our simulated data, we have 'Boiler', 'Furnace', and 'HeatPumps' for heating type, 'HardWood' and 'Ceramic' for floor type, 'Honey', 'WhiteAshed', 'Brown' for floor color, 'Wool', 'Polyester' and 'Olefin' for carpet type, 'Cool', 'Warm', 'Bold' for carpet color. To implement this query, first, use a Common Table Expression (CTE) to select all apartments with review scores above the average. Then we use five subqueries to find the most frequently appear option in each of these five features. To find the most popular heating type, the Apartment and Building tables are inner joined.

The output in the Figure 13 shows the most frequently equipped property in each of the 5 different feature types. For example, the most popular floor color is honey, which can encourage more floor production with the color of honey.

	FloorType	FloorColor	CarpetType	CarpetColor	HeatingType
►	Ceramic	Honey	Wool	Cool	Furnace
*					

Figure 13: Popular Feature

7.3 Company Rank

The property management companies with review scores above the average are ranked in average review score. This query gives us clues about which companies have better reputations in the aspect of their apartment quality.

First, the overall average review score is found by using subquery and declared as a variable. Then the Owners, Buildings, Apartments tables are inner joined to find the companies whose average review score of their apartments are above the overall average review score. The windows function is used to give a rank to these companies in order of their average review scores.

The output is a list of companies ranked in order of average review score. The Figure 14 only shows parts of the results, more results can be found in the application interface.

7.4 Luxury Apartments

If we consider an apartment with a washer and dryer in the unit, as well as a gym and pool in its building as a luxury apartment, we can compare the number of luxury apartments in different cities.

First, we use two subqueries to find all luxury apartments, which is, to select ApartmentID in the ApartmentFeature or BuildingFeature table which meets the requirements. Then the luxury apartments are counted based on cities.

The output is a list of the number of luxury apartments in different cities (Figure 15). From the result, Haysville has the most number of luxury apartments, which may not be the case in real life. This is because the data is randomly generated. We can expect that Kansas City may have the most number if the data is populated with real data.

	Rank	Name	AvgReviewScores
▶	1	Ajent	5
	1	ArrowCross	5
	1	Bud_Bloom	5
	1	CalciteX	5
	1	ClearMode	5
	1	CranialSpace	5
	1	Drifty	5
	1	FilterFlow	5
	1	Floria	5
	1	FlowFreer	5
	1	Flowz	5
	1	FoodiePhilic	5
	1	Humman	5
	1	LYTEiddle	5
	1	Newly	5
	1	Priorize	5
	1	ReelLite	5
	1	Seer	5
	1	Varietex	5

Figure 14: Company Rank

8 Summary and Discussion

The project is fully completed as our plan.

There are lots of changes since the proposal. The database design is improved by 1.merge Streets table into Buildings table. 2. Separate some features from Apartment and Building tables to create FeatureA and FeatureB tables and use bridge tables to connect the many-to-many relationships. 3. We also add a bridge table between the Apartment and Customers tables due to the many-to-many relationship. 4. Surrogate key and features like StartDate and EndDate, in ResideRecords table are used to allow customers to move out then move in.

What we learn from the project is that when making a plan or proposal, we should be more realistic but not too ambitious. At first, we hope our application is nation-wide, which we found it's hard to populate the database. So we change the choice of cities from cities in the U.S. to cities in Kansas. Also, since we hope to practice SQL coding, we want to have some delicate and complex design of report queries. But even we simplified the query design by the suggestions of our instructor, some queries are still hard to implement. Therefore, due to the limitation and also the convenience of implementation, some report queries are changed. For example, the third report query which was to rank the companies based on the review scores and also average month rent, we now change to rank the companies only on review scores. In the fourth report query, we count the number of luxury apartments instead of computing the luxury apartment proportion.

If we could redo this project, we hope to spend more time on the interface design. Because we have a complicated system, we hope we could have a more fluent experience when exploring the system. For example, we don't have a sign-in in the main page which can directly jump to the customer page without going through the search page. Now we learn the lessons and would be more careful and have more experience in interface design.

For future work, this project has the potential for further expansion. 1. If this database is populated with data in the real world, the report query will become much more meaningful and useful. 2. If we can collect coordinates information of the building, then we can locate the apartment on a map shown in the interface. 3. We can incorporate pictures of apartments into the database, which will be a powerful reference for customers. 4. We can expand the database

	NumApt	City
▶	46	Haysville
	44	Emporia
	38	Kansas_city
	35	Dodge_city
	25	Garden_city
	25	Olathe
	4	Lawrence
	3	Salina
	2	Manhattan
*		

Figure 15: Luxury Apartments

by adding more information about the apartments, for example, the amenity information.

9 Team Responsibilities

The responsibilities are assigned as following:

Da chan: database population, system design, and interface implementation.

Yijun Lin: data generation, data modeling, and data processing.

Yang Yang: SQL question and report queries, proposal, report, and slides.

All: database design, query design, interface design, team presentation.