

Property search application (e.g., residence, commercial space, land) for purchase/rental

COURSE: DATABASES (ECE_FK703)

Group 43

Ntounis Konstantinos

Stefas Filippou

1 SUMMARY

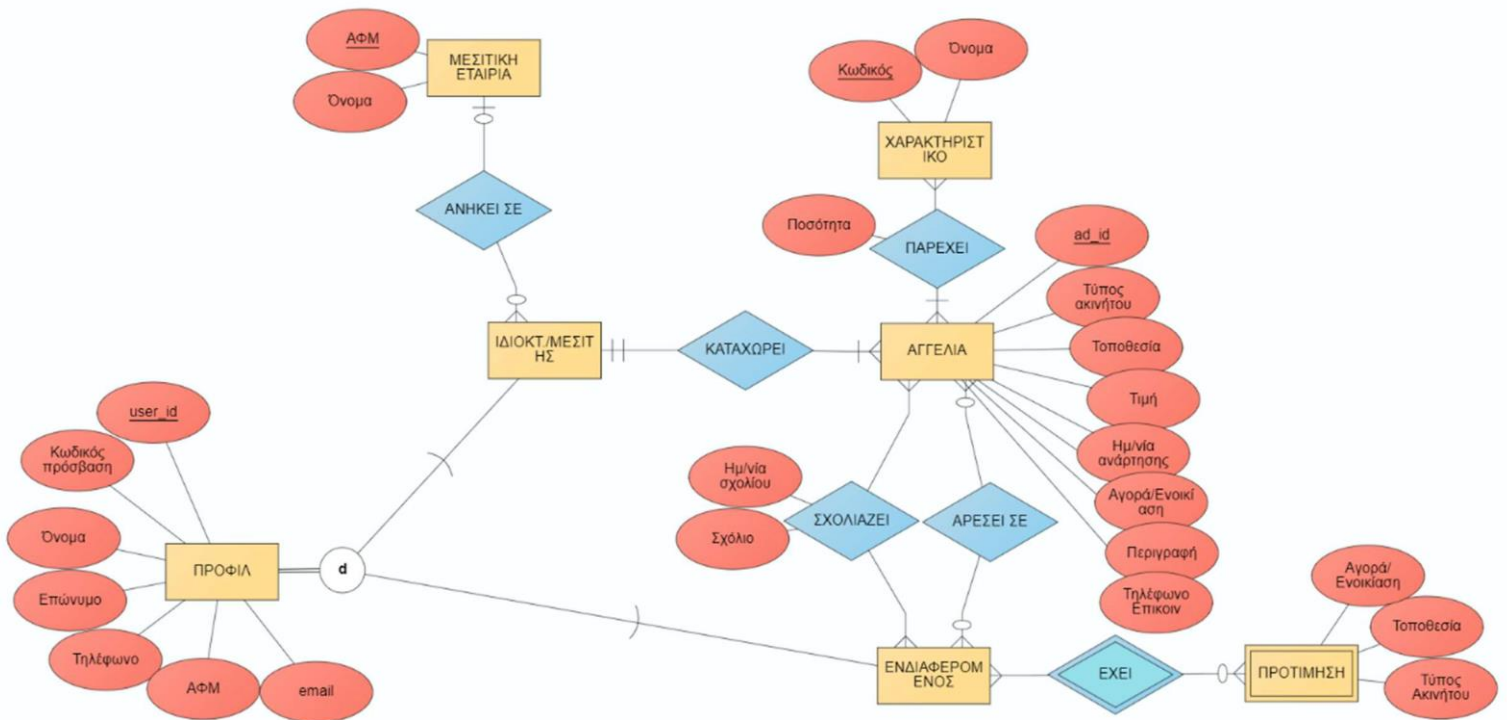
As part of the course we were assigned to implement a property search application. The basic idea of the microworld we created is that both interested parties and owners/brokers can create a profile and search or post an ad in the application. From the basic idea, it seems that our goal was to create a realistic, but also a two-way application, that is, one that is not only for those looking for a house, commercial space or land, but also for those who are interested in renting or selling a property.

2 METHODOLOGY, DATA COLLECTION AND TIMETABLE

When we were assigned this topic, we first looked at similar real estate applications on the internet and studied their structure and function. At the same time, we started creating our microcosm through an ERD, making the following assumptions:

- Ads and their Features, Users and Comments have numbers that uniquely identify them.
- Anyone interested in using the app is invited to create a profile.
- Each user can be either an owner/broker and post an ad, or an interested party, who filters some ads based on his/her preferences.
- Each advertisement lists the amenities available in each property.
- Each user can leave a comment below the posted ads.
- The broker may be owned by a real estate company.

Based on the above, we used ERDMakerto build the extended entity-relationship diagram. After gradual corrections and discussion with the teachers during the mid-term presentation (8/11), we came up with the following model:



Next, we worked on creating the corresponding relational model using DB Designer. We notice in the image below that in addition to the entities, we also created the "PROVIDE" table for the corresponding relationship. We also created, for the relationships "COMMENTS" and "LIKES" the tables "COMMENT" and "FAVOURITES" respectively, because they are of M:N type.

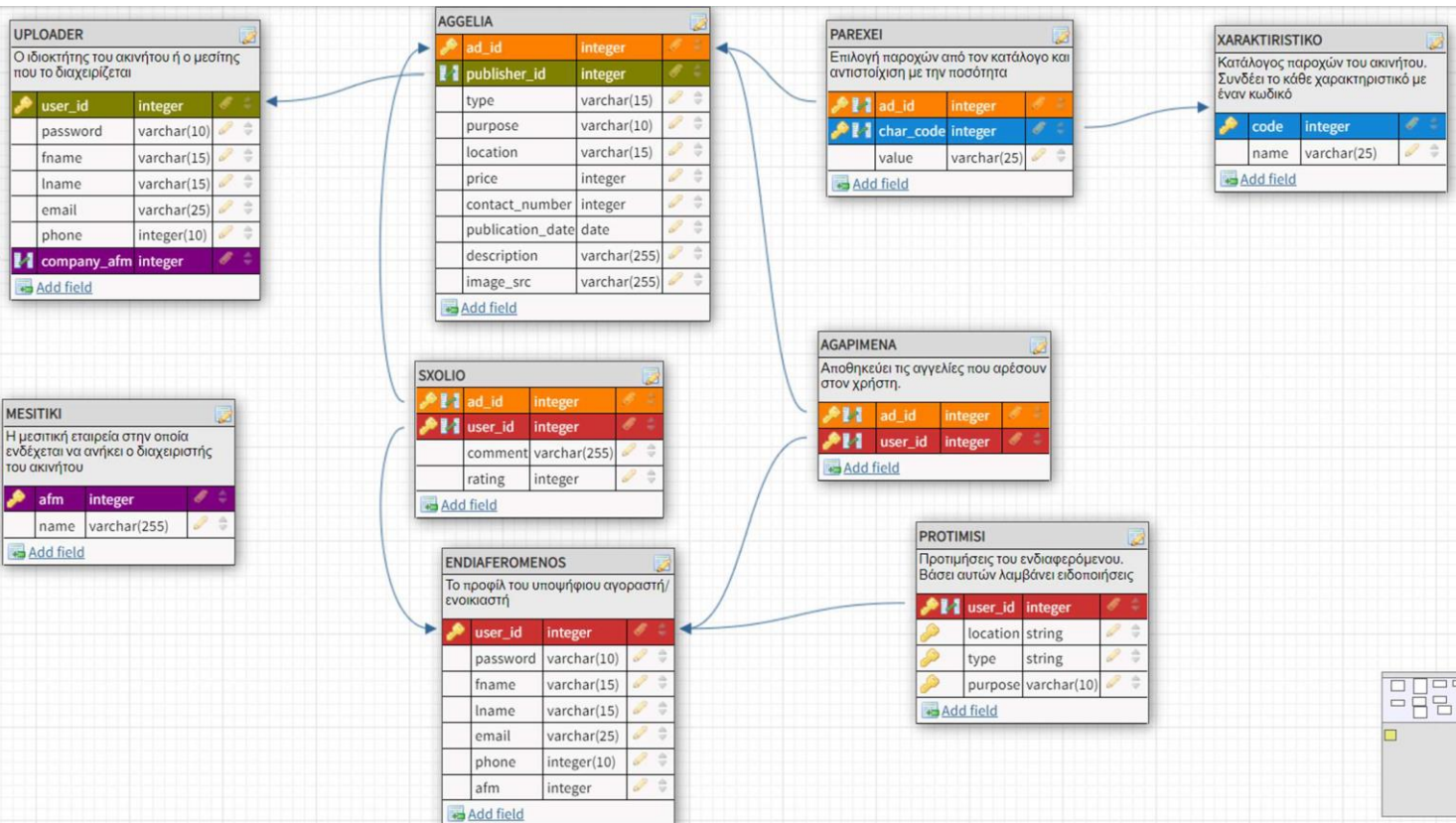


Figure 2: corresponding logical relational model

By the end of November we had completed Phase A of the project, which is the design of the ERD and the corresponding relational schema, and came up with how to manage the rest of the project.

In December, we created the database by executing the SQLite commands found in the create_tables.sql file we compiled. Using the python library sqlite3, we achieved communication between the application and the database. Therefore, by running the create_tables.py file, the database is created that is still empty.

Then we built create_ads.py, create_users_csv.py, create_comm_fav_prot.py, which create for each table .csv files with test data to be loaded into the database via insert_data.py. In create_ads.py we insert the ads and their benefits using the requests library. More specifically, we make requests to the xe.gr site receiving json files with the attributes of each ad. We filter these by extracting the information we want and create the .csv of the ads and their details. In the other 2 create files we create random information. In the create_users_csv.py file we take random names generated from the python 'names' library and assign the rest of the data to them in a random way, such as the phone number and whether the user in question belongs to a real estate company.

Similarly, we randomly assigned both preferences and favorites to each person and comments to each ad.

Finally, during the break of the academic semester, the functions of adding, changing, deleting and searching records of the main tables were tested first (in the SQLite DB Browser) and then implemented in the user interface (using the parameters chosen by the user):

AVAILABLE QUERIES

1. AD SEARCH WITH DESCRIPTION

e.g. `SELECT * FROM AGGELIA WHERE title LIKE '%key%' OR description LIKE '%key%';`


2. LOCATION FILTERS - PURPOSE - TYPE - CHARACTERISTICS (E.G. SQUARE FOOTAGE, ROOMS)

e.g. `SELECT AGGELIA.ad_id, AGGELIA.price FROM AGGELIA, PAREXEI AS P1, PAREXEI AS P2
WHERE location='Patra' AND type='residence' AND purpose='rent'
AND (price BETWEEN {min_price} AND {max_price})
AND AGGELIA.ad_id=P1.ad_id AND P2.ad_id=AGGELIA.ad_id
AND P2.char_code=2 AND P2.value='1' AND P1.char_code=1;`

3. FINDING ADS THAT MATCH THE USER BY NUMBER

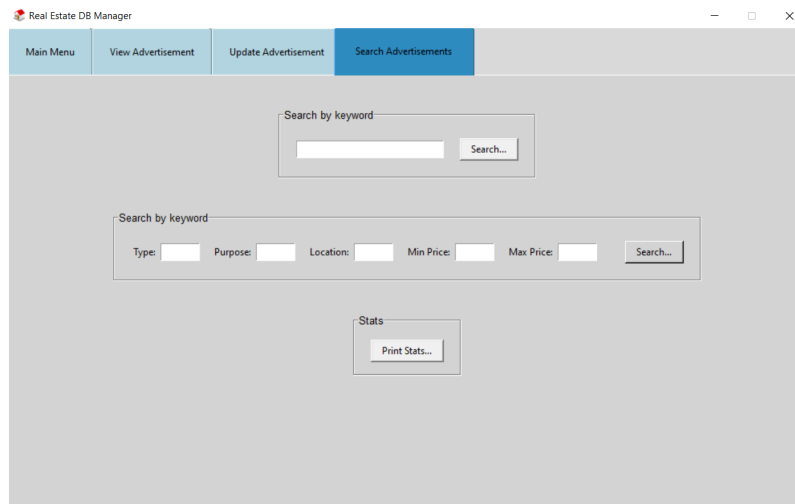
e.g. `SELECT ad_id FROM AGGELIA NATURAL JOIN PROTIMISI WHERE user_id=123;`

4. PRICE STATISTICS GROUPED BY TYPE AND PURPOSE OF REAL ESTATE AD



Ad_id	Title	Location	Purpose	Price	Contact Number
1	Διαμέρισμα 78 τ.μ.	Πάτρα	rent	4000 €	6900000000
2	Διαμέρισμα 58 τ.μ.	Πάτρα	rent	580 €	6900000000
4	Διαμέρισμα 80 τ.μ.	Πάτρα	rent	750 €	6900000000
5	Διαμέρισμα 100 τ.μ.	Πάτρα	rent	900 €	6900000000
7	Διαμέρισμα 32 τ.μ.	Πάτρα	rent	400 €	6900000000
10	Διαμέρισμα 52 τ.μ.	Πάτρα	rent	450 €	6900000000
12	Διαμέρισμα 28 τ.μ.	Πάτρα	rent	280 €	6900000000
14	Διαμέρισμα 39 τ.μ.	Πάτρα	rent	240 €	6900000000
15	Διαμέρισμα 35 τ.μ.	Πάτρα	rent	250 €	6900000000
20	Διαμέρισμα 97 τ.μ.	Πάτρα	rent	400 €	6900000000
21	Διαμέρισμα 60 τ.μ.	Πάτρα	rent	230 €	6900000000
24	Διαμέρισμα 50 τ.μ.	Πάτρα	rent	250 €	6900000000
25	Διαμέρισμα 115 τ.μ.	Πάτρα	rent	500 €	6900000000
27	Διαμέρισμα 45 τ.μ.	Πάτρα	rent	350 €	6900000000
28	Διαμέρισμα 60 τ.μ.	Πάτρα	rent	280 €	6900000000

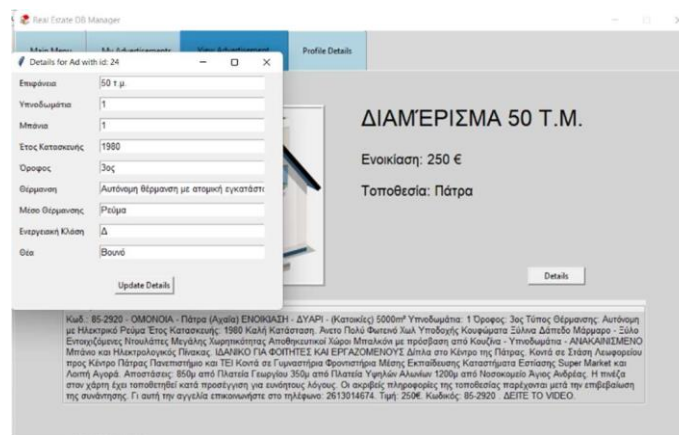
Action Buttons



ADVERTISEMENT MANAGEMENT

1. SELECTION AND MODIFICATION OF AD ELEMENTS AND PROVIDERS

e.g. UPDATE AGGELIA SET title='NEW TITLE' WHERE ad_id=123;



POSSIBLE FUNCTIONS

1. PRICE DROP NOTIFICATION AT EVERY PRICE DROP FOR FAVORITE ADS
2. AD DISPLAY SELECTED WITH COMMENTS AND AVERAGE RATING
3. PRICE STATS DISPLAY FOR EVERY TYPE OF REAL ESTATE

3 EVALUATION

After the demonstration of the application, we see that it is functional and quite realistic. All the details that a user needs when searching for a property are provided. There are no errors presented to the user when running it.

Furthermore, the average user would find it difficult to handle the program from the python command line. Therefore, with the help of the tkinter library we created the graphical interface of our application, as we thought that this would make it even easier to use.

4 MAIN ACTIONS

The individual tasks that created the present result were divided as follows:

- After studying the issues and other available implementations and discussion, the ERD and Relational schema diagrams were jointly organized.
- Also collaborative was the creation of the SQLite queries that were applied in our implementation and the data output entered.
- There was a differentiation in the continuation of the project, the student Konstantinos Dounis dealt with the creation of the graphical interface, while Philippos Stefas dealt with the writing of the report.

5 INSTALLATION INSTRUCTIONS

- i. Installing the pandas and pillow library (pip install ...)
- ii. Execute create_tables.py (Runs the SQL script to create tables)
- iii. Execute insert_data.py file
- iv. Execute gui.py file
- v. In the Login screen we use username admin - password admin to use all available administrator functions.

In python, the following basic libraries were used:

- Sqlite3: for communication with the database and execution of queries
- Pandas: for the import of data
- Tkinter: for creating the graphical environment
- Requests: for the request of ads
- Json: for processing requests

The link to GitHub: <https://github.com/KonstantinosN00/Real-Estate-Database>