Item Database in a House

Team 9

December, 2017

Abstract

The purpose of this document is to demonstrate the final implementation of our House Database System. The problem we have solved is finding a way to keep track of the location of items in a house. A house can have several different rooms across many floors. It can also have any number of different storage locations and items in each room. As a result it can be difficult to keep track of the location of different items as people move them around a house.

Our database solves this problem by creating a way to track items in a house. A house is divided into a number of floors depending on its size. Each of these floors is registered with a unique number into the database. Each of these floors can also have any number of unique rooms that can be identified by their unique room id numbers. Inside these rooms are storage locations that hold Items. Items can be stored in the database in either a room or storage locations. Additionally, Items can simply be left in rooms. They do not have to be in storage locations.

The database also registers family members as users to a specific house. A family member can query the database to find where an item is located as well as who owns the item. Family members can also move items in the house and update the location of these items.

This document demonstrates the ERD and UML diagrams outline in addition to the relations and their attributes. The ERD was mapped into the schema shown below in order to define each relation and its attributes. Furthermore, it was normalized to meet the requirements of our database.

Team members

Carlos Bustos, 1001317137

Junu Dhungana, 1001246500

Natnael Kebede, 1001149004

Octavio Garcia, 1000988989

Robert Leblanc, 1001001341

Table of Contents

Acronyms

ERD	Entity Relationship Diagram
UML	Unified Modeling Language

Project Documentation

Finding Items around a house can be a difficult task at times. A House has any number of locations that an Item could be located. That is across a number of rooms and floors. Items are also moved around the house by the family members residing in the house. As a result, there is no guarantee that an item will be where you left it or even where it should be located. Due to all of these factors, keeping track of sharable items as they move around the house can be a very difficult task. Our database system simplifies this process for family members.

The mission of the project is to simplify the process of locating an Item in a house. The database is capable of having multiple houses for different families. Each house is divided into different floors rooms and storage locations. Items are stored in rooms and storage locations. Family members can track the location of the items, who currently is using an item, and update the location of an item as they move it around the house.

The Database is capable of representing and grouping data using the following eight Relations: HOUSE, FLOOR, ROOM, STORAGE_LOCATION, ITEM, FAMILY_MEMBER, HOUSE_OF_FMBR, and ITEM_LOCATION.

The HOUSE Relation represents a house that a family lives in and contains two attributes. The House_id which is a unique identifier for each house, is a char of length ten that cannot be NULL. The House_id is the primary key of the House relation. The second attribute is Addr. It contains the address of the house and is a char of length sixty that cannot be NULL.

The FLOOR Relation represents the floors in each house and has three attributes. Floor_numb which is the unique identifier for each floor in a house, is a char of length two that cannot be NULL and is the primary key of the Relation. The Second attribute of the relation is Description which contains a description of each floor. Description is a char of length sixty that

is defaulted to NULL unless otherwise specified. The third attribute of the Floor relation is House_id which contains the unique identifier for the house that the floor is located in. House_id is a char of length ten that cannot be NULL and is a foreign key that points to the House_id in the House relation.

The ROOM relation represents the rooms in each house and has five attributes. The first attribute is Room_id which is the unique identifier for each room in a house. Room_id is a char of length two that cannot be NULL and is the primary key of the Room Relation. The second attribute is Description which provides a brief description of the room. Description is a char of length sixty that is defaulted to NULL if no value is specified. The third Attribute is Room_owner. It contains the unique identifier for the family member that owns the room. Room_owner is a char of length two that cannot be NULL. It is also a foreign key that points to the Family_id in the Family member relation. The fourth attribute is Floor_num which contains the unique identifier of the floor the room is located on. Floor_numb is a char of length two that cannot be NULL. It is also a foreign key that points to the Floor_numb in the Floor relation. The fifth attribute is House_id which is the unique identifier for the house that the room is in. House_id is a char of length ten that cannot be NULL. It is a foreign key that points to the House_id in the House Relation.

The STORAGE relation represents storage locations that items can be placed in and have seven attributes. The first Attribute is Storage_id which contains the unique identifier for the storage location. Storage_id is a char of length two that cannot be NULL. Storage_id is the Primary key of the storage Relation. The Second attribute is Storage_capacity which specifies the maximum number of items that can be stored in a given location. Storage_capacity is a varchar of length two that is defaulted to NULL. The third attribute is Description which contains a description of the storage location. Description is a char of length sixty that is

defaulted to NULL. The fourth attribute is Room_id which contains the unique identifier for the room the storage Location is located in. Room_id is a char of length two that cannot be NULL. Room_id is also foreign key that points to the Room_id in the Room Relation. The fifth attribute is the Room_owner which contains the unique identifier for the family member that owns the Room. Room_owner is a char of length two that cannot be NULL. Room_owner is a foreign key that points to Family_id in the Family member Relation. The sixth attribute is Floor_num which contains the unique identifier for the floor the storage location is on. Floor_num is a char of length two that cannot be NULL. Floor_num is also a foreign key that points to the Floor_num in the Floor Relation. The seventh attribute is House_id which contains the unique identifier for the House that the storage unit is in. House_id is a char of length 10 that cannot be NULL. It is also a foreign key that points to the House_id in the House Relation.

The ITEM relation represents items that are in a house and has five attributes. The first attribute is Item_id which is the unique identifier for each item in a house. Item_id is a char of length two that cannot be NULL. It is the primary key of the Item relation. The second attribute is the Placed_date attribute which tells you when the item was placed in a given location. Placed_date is a date that cannot be NULL. The third attribute is Removed_date which represents the day an Item was removed from a house. Removed_date is a date that cannot be NULL. The fourth attribute is Description which provides a description of the item. Description is a char of length sixty that is defaulted to NULL. The fifth attribute is Item_owner which represents the unique identifier for the family member who owns the item. Item_owner is a char of length two that cannot be NULL. It is a foreign key that points to Family_id in the FAMILY_MEMBER relation.

The FAMILY_MEMBER relation represents the family members who live in a house and has seven attributes. The first attribute is Family_id which represents the unique identifier

for a given family member. Family_id is a char of length two that cannot be NULL. Family_id is the primary key of the FAMILY_MEMBER Relation. The second attribute is F_name which represents the first name of a given family member. F_name is a varchar of length fifteen which cannot be NULL. The third attribute is M_init which represents the middle initial of a family member. M_init is a char of length one that is defaulted to NULL. The fourth attribute is L_name which represents a family member's last name. L_name is a varchar of length fifteen that cannot be NULL. The fifth attribute is Age and represents the age of a family member. Age is an int that cannot be NULL. The sixth attribute is Sex which represents the sex of a family member. Sex is a char of length one that cannot be NULL. The seventh attribute is Birthday which represents the birthdate of a family member. It is a data type of date and cannot be NULL.

The HOUSE_OF_FMBR relation represents which family member owns which house and contains two attributes. The first attribute is Family_id which represents the unique identifier of the family member that owns the house. Family_id is a char of length two that cannot be NULL. It is a foreign key that points to Family_id in the FAMILY_MEMBER Relation. The second attribute is House_id which represents the unique identifier for a house that is owned by a family member. House_id is a char of length ten that cannot be NULL. It is a foreign key that points to House_id in the House Relation.

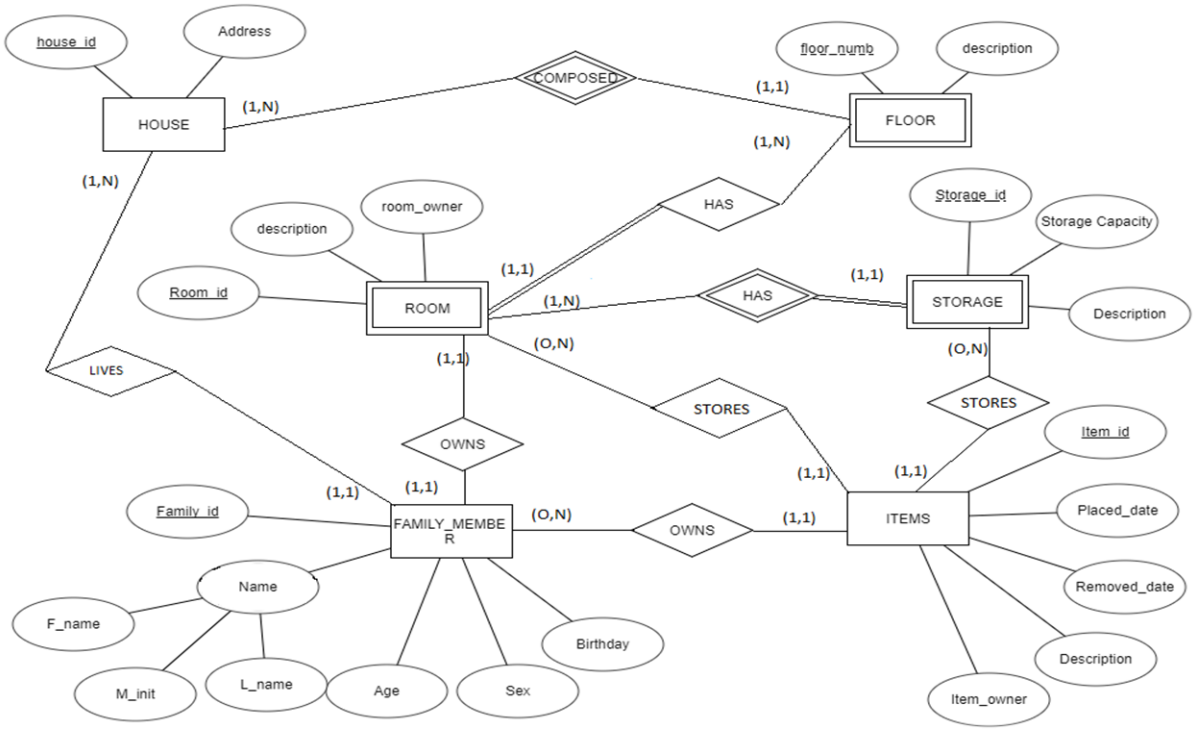
The ITEM_LOCATION relation represents where each item is located in a house and has seven attributes. The first attribute is Item_id which represents the unique identifier for each item in a house. Item_id is a char of length two that cannot be NULL. It is a foreign key that points to Item_id in the item Relation. The second attribute is Item_owner which contains the unique identifier for the family member who owns the item. Item_owner is a char of length two that cannot be NULL. Item_owner is a foreign key that points to the Family_id in the FAMILY_MEMBER Relation. The third attribute is Room_id which contains the unique

identifier for a room in a house. Room_id is a char of length two that cannot be NULL. Room_id is a foreign key that points to the Room_id in the ROOM relation. The fourth attribute is Room_owner which contains the unique identifier for the family member that owns the room an item is located in. Room_owner is a char of length two that cannot be NULL. It is a foreign key that points to the Family_id in the FAMILY_MEMBER Relation. The fifth attribute is Floor_num which contains the unique identifier for the floor the item is stored on. Floor_numb is a char of length two that cannot be NULL. Floor_num is a foreign key that points to Floor_num in the floor relation. The sixth attribute is House_id which contains the unique identifier for the house that the item is stored in. House_id is a char of length ten that cannot be NULL. House_id is a foreign key that points to House_id in the house Relation. The seventh attribute is Storage_id which contains the unique identifier of the storage location the item is located in. Storage_id is a char of length two and cannot be NULL. It is a foreign key that points to Storage_id in the storage Relation.

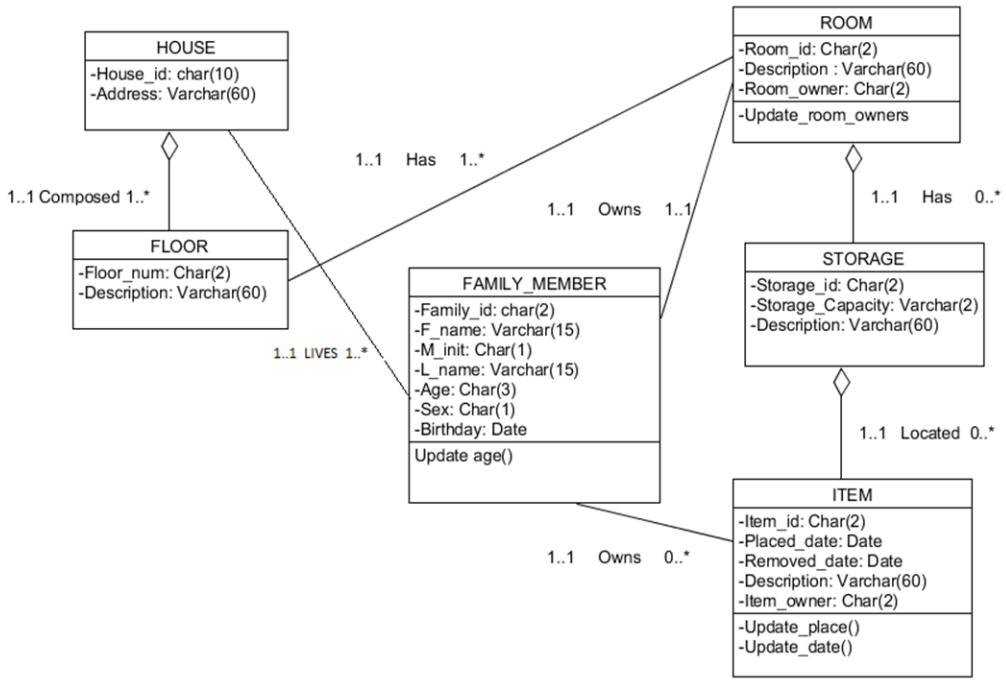
The Database has the following Requirements. Every Item must be in a storage location. Items cannot exist outside of a storage location. Any Id can function as a storage Id such as a room or a floor Id. The items history will not be tracked nor will the dates of the item placed be updated. The database will simply track and update the Location of the items in the house. If an Item is removed from the house, the item will only display a removed date.

For the Database we assume that an attic or basement is its own floor and allows the storing of items. We also assume that there are no storage locations within storage locations. For example there would not be a box of items inside a desk drawer. All of the items would be in the one location.

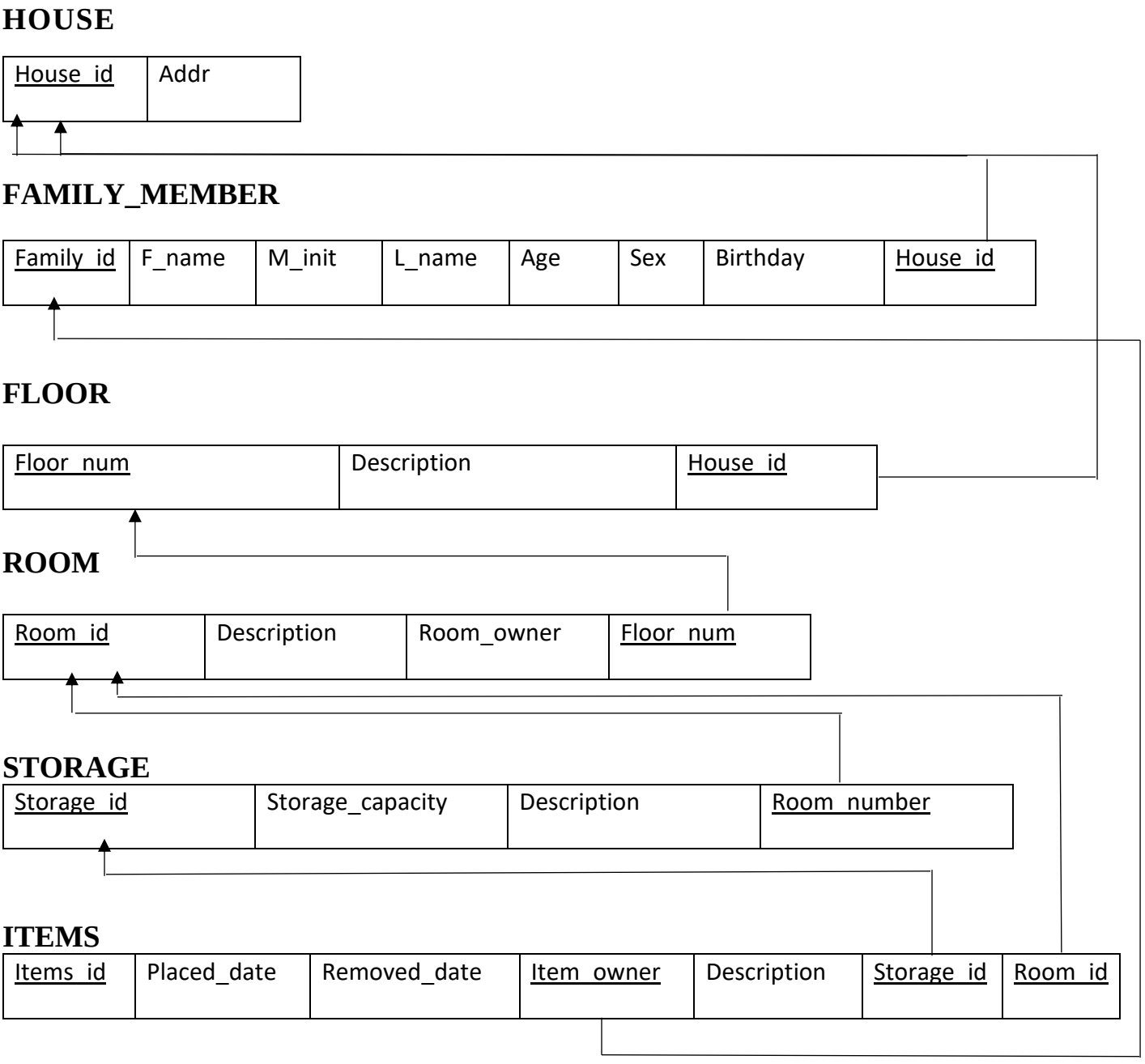
Updated Entity Relationship Diagram (ERD)



Unified Modeling Language (UML)



Result of Mapping the House ERD Schema into a Relational Database Schema



Possible Queries for the database

Query 1:

Demonstrate a query that will show all items owned by an individual person.

- To do this command, we have to use the ITEM and FAMILY_MEMBER relation. Using the Item_owner and Family_id we could filter which item belong to which individual.

SQL command:

```
SELECT item.Description, family_member.F_name, family_member.L_name
FROM item
JOIN family_member ON
Item.Item_owner = family_member.Family_id
```

SQL output:

Description	F_name	L_name
Computer	Carlos	Bustos
Watch	Junu	Dhungana
Belt	Natnael	Kebede
Tshirt	Robert	LeBlanc
Teddy Bear	Octavio	Garcia
Sweater	Natnael	Kebede
Remote Control Car	Octavio	Garcia

Query 2:

Demonstrate where all the items are in any house.

- By taking into account the Room_owner and Item_owner we are able to locate that item in our database.

SQL command:

```
SELECT room.Description , item.Description
FROM room
JOIN item ON
room.Room_owner = item.Item_owner;
```

SQL output:

Description	Description
Carlos' Room	Computer
Junu's Closet	Watch
Robert's study	Belt
Nate's living room	Tshirt
Octavio's garage	Teddy Bear
Robert's study	Sweater
Octavio's garage	Remote Control Car

Query 3:

Print out the address of all family members in the database with their names next to it.

- To do this command, we would need the HOUSE_OF_FMBR, FAMILY_MEMBERS and house relations. Each member in a family has a House_id which identifies where that individual lives. Comparing this unique identifier, we can get their address.

SQL command:

```
SELECT house.addr , family_member.F_name , family_member.L_name
FROM house , family_member, house_of_fmbr
WHERE house.house_id = house_of_fmbr.House_id
AND house_of_fmbr.Family_id = family_member.Family_id
GROUP BY family_member.L_name
ORDER BY family_member.Family_id;
```

SQL output:

addr	F_name	L_name
100 Oak Hollow Ct. S Azle Tx 76020	Carlos	Bustos
1008 Wester Trl. Keller Tx 76248	Junu	Dhungana
100 River Creset Ct. Aledo Tx 76008	Natnael	Kebede
10020 Tule Lake Rd. Ft. Worth Tx 76177	Robert	LeBlanc
100 Spur Ct. Aledo Tx 76008	Octavio	Garcia

Query 4:

Demonstrate what items are owned by any female in our database.

- Using the ITEM and FAMILY_MEMBER relation we are able to filter out all family members who are not and compare their Family_id with the item's owner.

SQL command:

```
SELECT item.Description , family_member.F_name , family_member.M_init ,
family_member.L_name
FROM item , family_member
WHERE item.Item_owner = family_member.Family_id AND family_member.Sex = 'F'
GROUP BY family_member.F_nameORDER BY item.Description
```

SQL output:

Description	F_name	M_init	L_name
Watch	Junu		Dhungana

Query 5:

Show all the family members who are under the age of 25.

- Just as query 4, this query is filtering. It needs the use of one relations since we do not want to print out anything besides the family members' first and last name.

SQL command:

```
SELECT family_member.Age , family_member.F_name , family_member.M_init ,
family_member.L_name
FROM family_member
GROUP BY family_member.Age
HAVING family_member.Age > '25'
```

SQL output:

Age	F_name	M_init	L_name
26	Junu		Dhungana
28	Octavio		Garcia

Appendix A

Purpose of this project

We are in the twenty-first century and everything around us is being computerized, so why not our daily activities? We have more important things to do than search for items in a house. We need a software that can help us manage sharable and frequently used items in any house. The purpose of this project is to create a database system that can help people find sharable items in their house.

It is very difficult to keep track of items that are sharable by family members whenever they are need by the others. This database system will help you fix your problem by tracking the items: where the item was placed, when the item was placed, and who placed the item (Presentation, p. 2).

Appendix B

Technical words

Database: collection of data in certain way that is easier for application software to find the desired data quickly. Our database system has a collection of houses, family members from each houses, number of rooms in the house, number of floor, other kinds of storage items in that house and sharable items in the houses. It can keep track of the sharable items in any house along with the time that was placed in certain location and when it was removed form that location. Our database can also be queried, defined, and updated as per the user's requirement changes (Project.pdf, p.2).

UML: Unified Modeling Language (Presentation, p. 8 - 10). Created using a software called UMLet.

ERD: Entity Relationship Diagram (Presentation, p. 9 - 10). Created using software called ERD plus.

Mapping: This is a mechanism of showing how two entities are related with each other (Project.pdf, p.13).

Schema: Schema in database defines how the data is organized. in our case it is organized in the form of tables (Project.pdf, p. 13).

Relational Database: This is a set of tables with data collected in a certain manner (Project.pdf, p. 13). Here, we have six relations: House, FAMILY_MEMBER, FLOOR, ROOM, STORAGE and ITEMS.

Normalizing: This is the process of organizing data in our database in order to avoid any kind of data redundancy such as insertion, deletion, and update abnormalities. This is done after the mapping phase. We performed three types of normalization in order to avoid data redundancy

and these steps were the first normal form (1NF), the second normal form (2NF), and the third normal form (3NF) (Presentation, p. 12).

Populating the Database: This requires filling the database with dummy data in order to check that the database system we just created works properly or not. We populated each of our relations with real world data (Presentation, p. 14 - 21).

Querying the database: This is a way of extracting data from the database using SELECT statement in any database management system (Presentation, p. 23 - 27).

Appendix C

Design

To design the database and to get the concept about how this database should work, we first created UML and then we created ERD. After that Mapping was done and it was followed by Normalization. After that, we populated the data and then queried the database in order to check if the database is working as it is supposed to do.

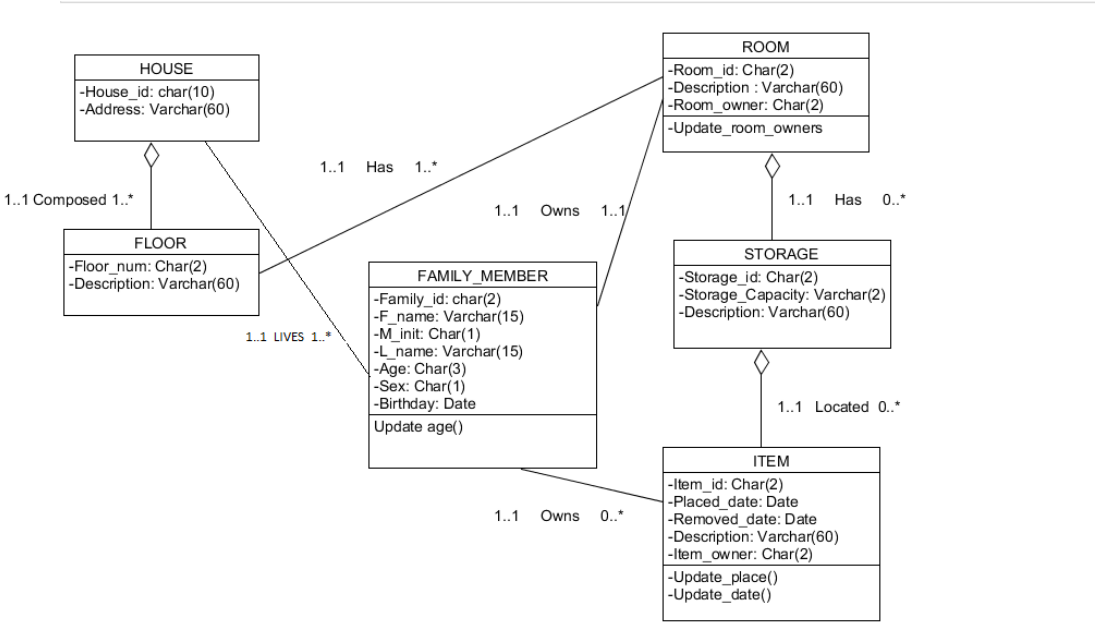


Figure 1. This figure illustrates the UML we created for our database. It is the representation of how our SQL database is going to be in phpMyAdmin (Presentation, p. 8).

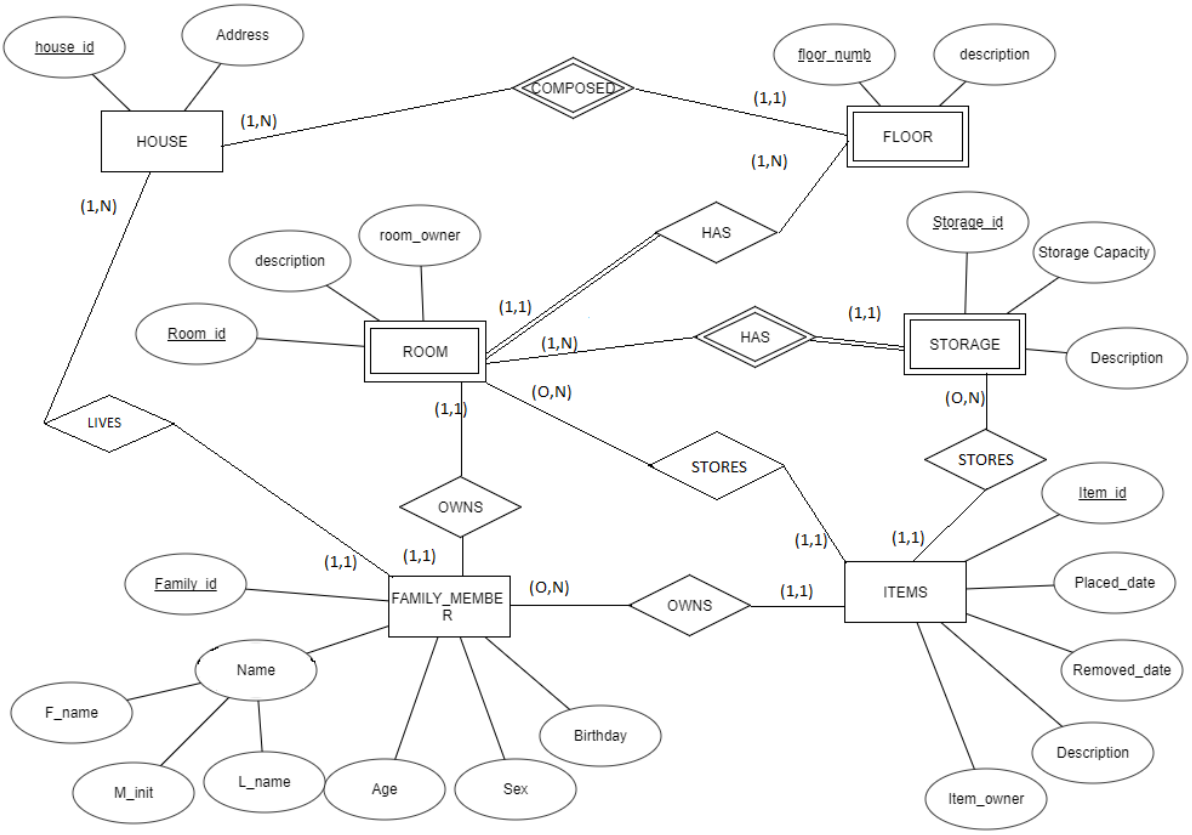


Figure 2: This figure shows our actual ERD for this database. It is pictorial representation of our database (Presentation, p. 9).

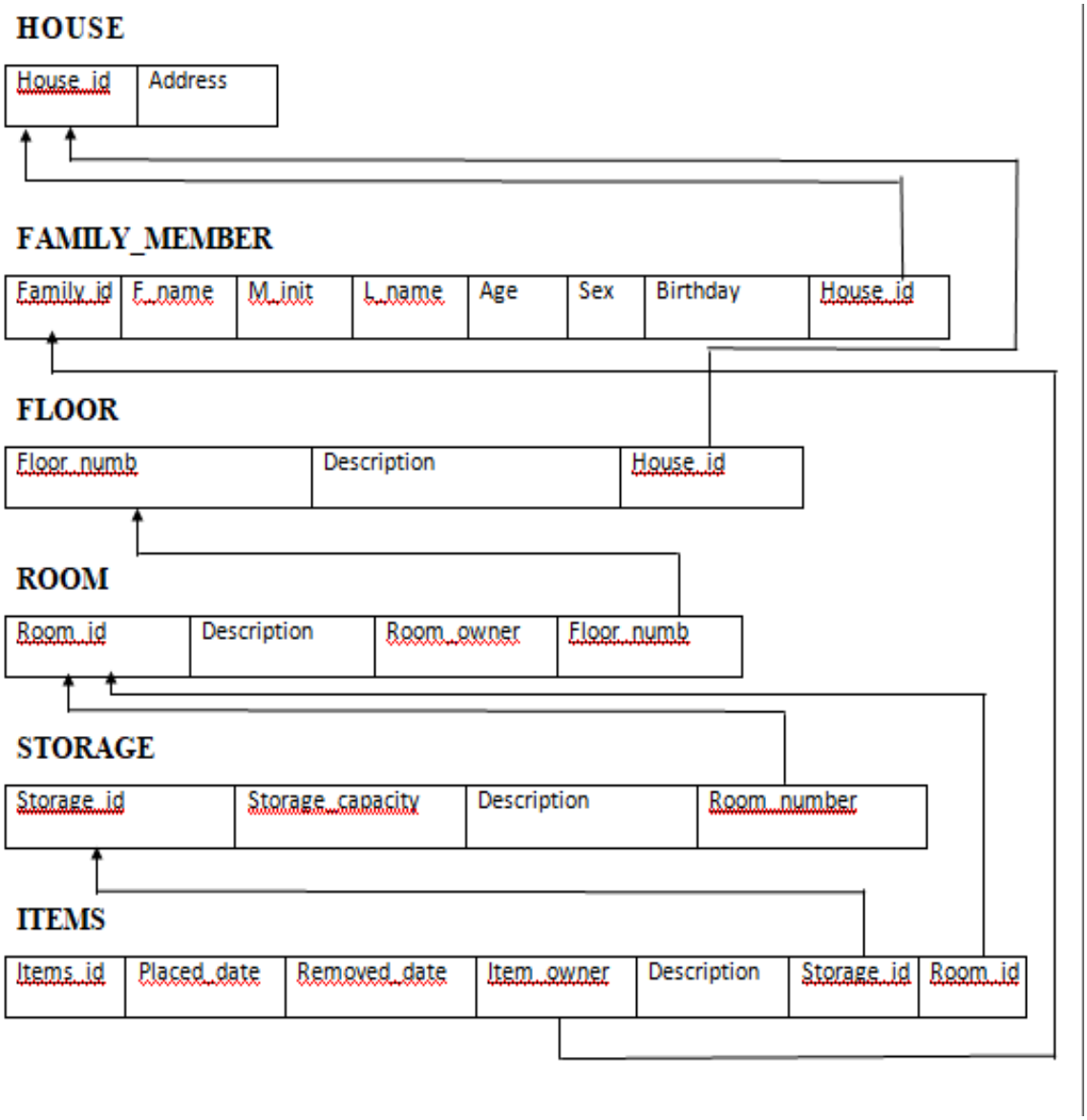


Figure 3. This figure is result of mapping the House ERD schema into a Relational database schema (Project.pdf, p. 13).

Appendix D

Implementation of Database

	Family_id	F_name	M_init	L_name	Age	Sex	Birthdate
Edit Copy Delete	1	Carlos	E	Bustos	22	M	1995-06-24
Edit Copy Delete	2	Junu		Dhungana	26	F	1991-02-06
Edit Copy Delete	3	Natnael	K	Kebede	22	M	1996-10-31
Edit Copy Delete	4	Robert		LeBlanc	22	m	1995-07-04
Edit Copy Delete	5	Octavio		Garcia	28	m	1988-12-31

Figure 4. This figure shows the data we populated for the FAMILY_MEMBERS relation.

	house_id	addr
Edit Copy Delete	249000	100 Oak Hollow Ct. S Azle Tx 76020
Edit Copy Delete	265000	1008 Wester Trl. Keller Tx 76248
Edit Copy Delete	325995	10020 Tule Lake Rd. Ft. Worth Tx 76177
Edit Copy Delete	333900	100 Spur Ct. Aledo Tx 76008

Figure 5. This figure shows the data we populated for the HOUSE relation.

	Floor_numb	Description	House_id
Edit Copy Delete	1	First Floor	249000
Edit Copy Delete	1	First Floor	265000
Edit Copy Delete	1	First Floor	325995
Edit Copy Delete	1	First Floor	333900
Edit Copy Delete	1	Basement	589999
Edit Copy Delete	2	Attic	249000
Edit Copy Delete	2	Attic	265000
Edit Copy Delete	2	Attic	325995
Edit Copy Delete	2	Attic	333900
Edit Copy Delete	2	First Floor	589999
Edit Copy Delete	3	Attic	589999

Figure 6. This figure shows the data we populated for relation FLOOR in our database.

	Storage_id	Storage_capacity	Description	Room_id	Room_owner	Floor_numb	House_id
Edit Copy Delete	1	5	Cabinet	1	1	1	249000
Edit Copy Delete	1	15	Moving box	1	1	2	249000
Edit Copy Delete	1	3	Nike's shoe Box	1	2	1	265000
Edit Copy Delete	1	10	Desk Drawer	1	3	1	589999

Figure 7. This figure shows the data we populated for the STORAGE relation in our database.

	Room_id	Description	Room_owner	Floor_numb	House_id
Edit Copy Delete	1	Carlos' Room	1	1	249000
Edit Copy Delete	1	Junu's Closet	2	1	265000
Edit Copy Delete	1	Robert's study	3	1	325995
Edit Copy Delete	1	Nate's living room	4	1	333900
Edit Copy Delete	1	Octavio's garage	5	1	589999

Figure 8. This figure shows the data we populated for the ROOM relation in our database.

	Item_id	Placed_date	Remove_date	Description	Item_owner
Edit Copy Delete	1	2017-01-01	2017-12-01	Computer	1
Edit Copy Delete	1	2017-09-01	0000-00-00	Watch	2
Edit Copy Delete	1	2016-05-02	0000-00-00	Belt	3
Edit Copy Delete	1	2017-03-15	0000-00-00	Tshirt	4
Edit Copy Delete	1	2017-02-14	0000-00-00	Teddy Bear	5
Edit Copy Delete	2	2016-05-10	2016-06-13	Sweater	3
Edit Copy Delete	2	2016-12-31	2017-07-04	Remote Control Car	5

Figure 9. This figure shows the data we populated for the ITEM relation in our database.

	Family_id	House_id
Edit Copy Delete	1	249000
Edit Copy Delete	2	265000
Edit Copy Delete	3	589999
Edit Copy Delete	4	325995
Edit Copy Delete	5	333900

Figure 10. This figure shows the data we populated for the HOUSE_OF_FMBR relation in our database.

	Item_id	Item_owner	Room_id	Room_owner	Floor_numb	House_id	Storage_id
Edit Copy Delete	1	1	1	1	1	249000	1
Edit Copy Delete	1	2	1	2	1	265000	1
Edit Copy Delete	1	3	1	3	1	589999	1
Edit Copy Delete	1	4	1	4	2	325995	1
Edit Copy Delete	1	5	1	5	2	333900	1
Edit Copy Delete	2	3	1	3	1	589999	1
Edit Copy Delete	2	5	1	5	1	333900	1

Figure 11. This figure shows the data we populated for the ITEM_LOCATION relation in our database.