DBMS Project Report

PES University

Database Management Systems

UE18CS252

Submitted By

PES2201800705 Supreet Ronad

The **Society Database Management System** is a Database System where all the required data related to the Societies, Houses, Residents, Visitors and Complaints are stored in a systematic way so that the reading or retrieval of any data becomes easier and efficient. All the assumptions made with regards to the Database are kept in mind and proceeded with the DATABASE design. The ER diagram contains the information about the Entity Types, Relationships and their Structural Constraints. The Logical Design where the Relational Schema is built along with the appropriate choice of Primary Keys and Foreign Keys for each Relation. The Physical Design which contains the DDL statements to create the database and its tables applying all the required constraints. A few Triggers and Retrieval Queries in SQL are also written to the Car Rental Database System.

The important part of the database is its efficiency considering the importance of the data stored in the database. We not only deal with the storing the data of the customers actually we also solved the issue of customers renting the car anywhere in any state without any restrictions imposed.

Index

Introduction

'Manual' managing activities has become outdated in the digital age. As we all are aware of the current housing society management system which is handled manually. The data is stored in the files and the processing of the data is done manually and the report generation is slow. Data which is required cannot be accessed quickly. The data is stored in various registers so the linking between it becomes difficult.

Technology has taken over every aspect of an individual's life, whether it is in the domestic sphere, at the workplace, or within the community. But this problem can be solved by using computerised database management systems. A perfect illustration of this is the management of residential societies/complexes with the aid of a relevant **Society Management System**. This is because the modern mindset demands more features, better administration, etc, in order to ensure that life is wonderfully hassle-free.

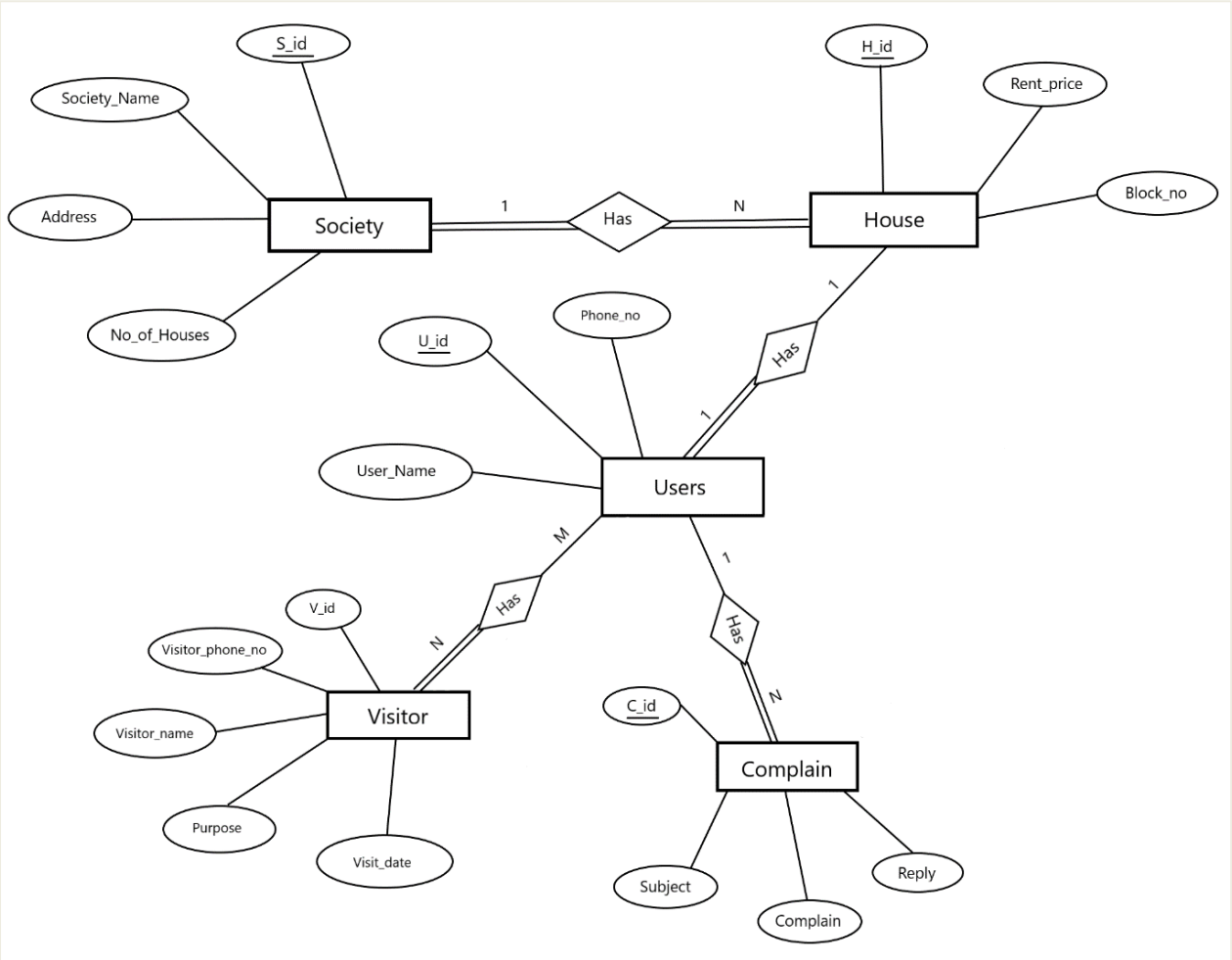
As the name of Database **Society Record Management System** suggests that in this system, we can store the details about all the required information about all the registered societies in a systematic way so that retrieval and storing of data, in future becomes easier and efficient. In this society management system all the society categorize by the number of houses. The people who lives in house they may be an owner of house or tenant of house. In this era, people are very busy with their routine work, so they do not have time for complain small problem related to houses. We have developed the system for society member they can make complain form anywhere any time and we resolve the complaint as soon as possible. In this system people can easily find address of the house by providing member name or id.

In our database there are a total of 6 tables including an extra table which keeps record of all the old info which was updated from Users table. Here **each user is given an user-id, each house a house-id, each society a society-id and complaints and visiting** also the same, which helps easy identification of any particular user house or anything related to a society. Hence the retrieval or reading and storing data becomes easier and faster.

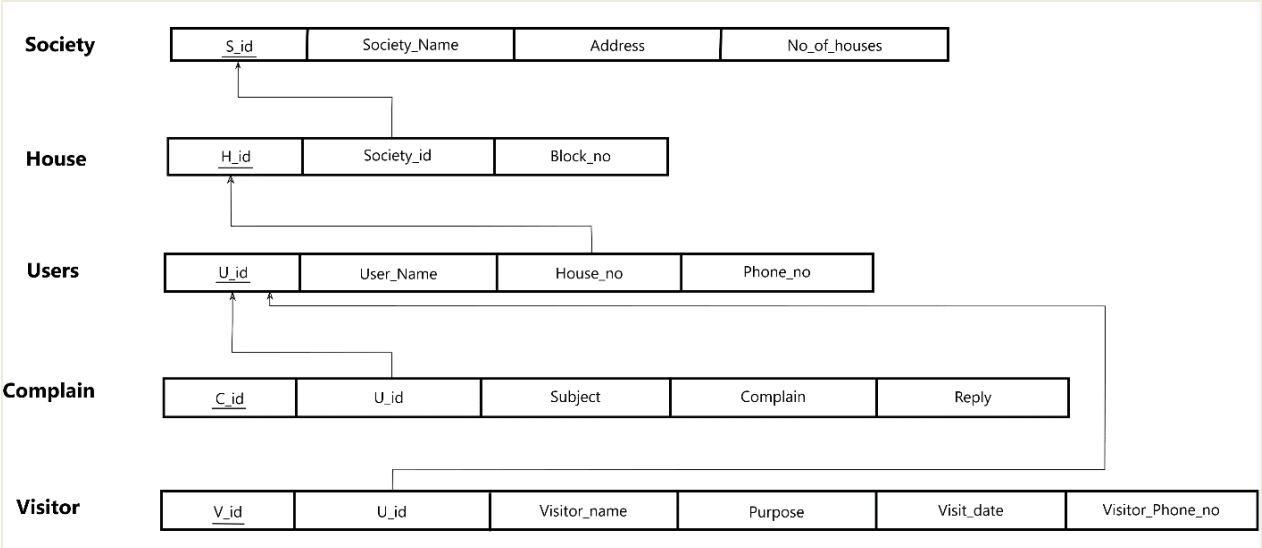
In the Society table the attributes are chosen such that two or more societies can have same name and location but, we can differentiate them based on S_id which is unique for each Society registered. Even for user and house unique id's are given. For House table, each house has a unique H_id and related informations like block_no, society_id and rent price are also added. For Users table, each user is given unique U_id . Apart from that each user's required information like Name, Phone number, House_no is also stored. Each user can send complain related to his house which will be stored in the Complain table, where user needs to enter the subject of the complain and the description about the complaint. Even the reply to the complain will be stored respectively. Visitors table has unique Visit id for each visits. It can also store visitor's name, phone number, date of visit and purpose. Time of the visit can also be added to make it more secure.

Data Model

ER Diagram



Schema



Functional Dependencies and Normalizations

Functional Dependencies

1. **Society** : S_id ->> {Society_Name, Address, No_of_Houses}
2. **House** : H_id ->> {Society_id, Block_no}
3. **Users** : U_id ->> {User_Name, House_no, Phone_no}
4. **Complain** : C_id ->> {U_id, Subject, Complain, Reply}
5. **Visitor** : V_id ->> {U_id, Visitor_name, Purpose, Visit_date, Visitor_Phone_no}

Primary Keys

1. S_id (Society)
2. H_id (House)
3. U_id (Users)
4. C_id (Complain)
5. V_id (Visitor)

Normalization

On converting the ER Model to the Relational Schema following all the rules of ER Model to Relational Schema mapping, the relations obtained are already in the normal form. However, under the following circumstances the relations may violate the Normal forms,

Consider adding the column 'Society_id' to the table 'Users', which gives information about the society in which the user lives. Upon adding,

Users {U_id, User_Name, House_no, Phone_no} **and**
U_id ->> {User_Name**, House_no**, Phone_no, **Society_id**}

But from House table we notice that , **H_id** ->> {**Society_id**, Block_no}

Here **Third Normal Form** is violated as this change causes transitive dependency among House_no, Society_id and User_id. In order to remove the violation, we decompose the relation into two relations ensuring the lossless - join property. The two relations are:

House {H_id, Society_id, Block_no} with **H_id** as Primary key **and Society_id** as Foreign key.

Users {U_id, User_Name, House_no, Phone_no} with **U_id** as Primary key and **House_no** as Foreign key.

Data Definition Language

CREATE DATABASE **Society**;
USE **Society**;

TABLE CREATION :

```
CREATE TABLE Society(
        S_id int not null,
        Society_Name varchar(24),
        Address varchar(100),
        No_of_Houses int,
        PRIMARY KEY (S_id)    );

CREATE TABLE House(
        H_id int not null,
        Society_id int not null,
        Block_no int,
        Rent_price int,
        PRIMARY KEY (H_id)    );

CREATE TABLE Users(
        U_id int not null,
        User_Name varchar(24),
        House_no int,
        Phone_no character(13),
        PRIMARY KEY (U_id)    );

CREATE TABLE Complain(
        C_id int not null,
        User_id int not null,
        Subject varchar(24),
        Complain varchar(200),
        Reply varchar(200),
        PRIMARY KEY (C_id)    );

CREATE TABLE Visitor(
        V_id int not null,
        U_id int not null,
        Visitor_name varchar(24) not null,
        Purpose varchar(24),
        Visitor_Phone_no character(13),
        Visit_date DATE not null,
        PRIMARY KEY (V_id)    );
```

ADDING CHECK AND REFERENTIAL INTEGRITY CONSTRAINTS :

ALTER TABLE House ADD CONSTRAINT soc_id FOREIGN KEY (Society_id) references Society (S_id) on delete cascade on update cascade;

ALTER TABLE Users ADD CONSTRAINT house_id FOREIGN KEY (House_no) references House (H_id) delete cascade on update cascade;

ALTER TABLE Complain ADD CONSTRAINT use_comp_id FOREIGN KEY (User_id) references Users (U_id) on delete cascade on update cascade;

ALTER TABLE Visitor ADD CONSTRAINT use_visit_id FOREIGN KEY (U_id) references Users (U_id) on delete cascade on update cascade;

```
ALTER TABLE Society
ADD CONSTRAINT CHECK (S_ID > 0 AND No_of_Houses > 0);

ALTER TABLE House
ADD CONSTRAINT CHECK (H_ID > 0 AND Block_no > 0);

ALTER TABLE Society
ADD CONSTRAINT CHECK (S_ID > 0 AND No_of_Houses > 0);

ALTER TABLE Users
MODIFY Phone_no character(13) not null;
```

EXAMPLES OF STATEMENTS FOR INSERTING VALUES INTO THE TABLES:

```
INSERT INTO Society VALUES (1,'Galaxy','HSR Layout,Banglore','75');
INSERT INTO Society VALUES (2,'Home Palace','Electronic city,Banglore','83');

INSERT INTO House VALUES (123,1,3,20000);
INSERT INTO House VALUES (255,2,1,25000);

INSERT INTO Users VALUES (705,'Supreet Ronad',123,'9686023454');
INSERT INTO Users VALUES (706,'Shashank G S',255,'7844023474');

INSERT INTO Complain VALUES (1024,705,'Water Problem','No Water available',null);
INSERT INTO Complain VALUES (1114,168,'Network Problem','No network','Done!');

INSERT INTO Visitor VALUES (445,068,'Anand Joshi','Meet',7089663467,'2020-08-02');
INSERT INTO Visitor VALUES (446,705,'Raam K','Courier',6723663897,'2020-08-02');
```

Populated Tables

Society Table

S_id	Society_Name	Address	No_of_Houses
1	Galaxy	HSR Layout,Banglore	75
2	Home Palace	Electronic city,Banglore	83
3	EcoStay	HSR Layout,Banglore	50
4	The Den	Banashankari,Banglore	115
NULL	NULL	NULL	NULL

House Table

U_id	User_Name	House_no	Phone_no
68	Anirudh Rajamouli	329	7777023488
168	Sneha Betageri	134	9211023478
333	Veeresh Panchaxari	239	8787873454
344	Jasmine Tased	512	9456783454
368	Nitish S	144	8888888444
522	Sathvik Nayaka	128	9688989894
552	Ayushi Agarwal	442	6666023414
554	Manjunath Hakki	205	6789023457
603	Shashi Rao	513	6996023454
632	Sapreetha Chittagubbi	729	8665833698
666	Pratyusha Jalabengi	506	8888023422
684	Sathvik Saya	216	6366023478
705	Supreet Ronad	123	9686023454
706	Shashank G S	255	7844023474
707	Sneha Chincholli	799	7786023884
711	Sanket Hiredesai	811	9681234564

Complaint Table

C_id	User_id	Subject	Complain	Reply
1002	554	Network Problem	No network coverage in my ap...	Network has been reset
1024	705	Water Problem	No Water available	NULL
1114	168	Network Problem	No network, please help	Done!
1156	168	Water Problem	Water not sufficient	Will be dealt with soon
1224	705	Electricity Problem	No power since yesterday, plea...	Will do
1227	68	Lift issue	No lights in lift, please fix	NULL
1234	554	Lift issue	Lift not working, please fix	NULL
1333	705	Noise Issue	Contruction is causing too muc...	Will look into that
1516	68	Electricity Problem	Daily an hour of power is being...	Nothing can be done about that
1523	554	Water Problem	No Water available	NULL
1578	852	Water Problem	No Water available	NULL
NULL	NULL	NULL	NULL	NULL

Users Table

H_id	Society_id	Block_no	Rent_price
239	4	4	27000
255	2	1	25000
259	2	1	23000
329	3	11	32000
442	3	8	21000
506	1	6	20000
512	2	8	19000
513	4	11	23000
722	1	10	14000
729	4	9	31000
799	4	9	18500
811	1	5	17500
1245	4	3	20000
NULL	NULL	NULL	NULL

Visitor Table

V_id	U_id	Visitor_name	Purpose	Visitor_Phone_no	Visit_date
345	705	Prem Chopra	Food Delivery	888967860	2020-11-13
405	852	Anand Joshi	Birthday Party	8888663469	2020-03-21
445	68	Anand Joshi	Just to meet	7089663467	2020-08-02
446	705	Raam Kalyan	Chit-chat	6723663897	2020-08-02
487	554	Rahul J	Food Delivery	8888666666	2020-10-10
547	554	Jake Peralta	Food Delivery	6942069420	2020-03-21
550	852	Jimbo Mckenzie	Birthday Party	7768970097	2020-03-21
689	552	Tahir T	Office meeting	8873456666	2020-04-15
785	705	Amy Santiago	Courier Service	7822367842	2020-04-13
885	705	Pam Beesly	Just to meet	8893453467	2020-03-21
4450	68	Anand Joshi	Just to meet	7089663467	2020-08-02
4451	68	Anand Joshi	Just to meet	3467	2020-08-02
NULL	NULL	NULL	NULL	NULL	NULL

Triggers

1. Trigger to store any update made to the Users Table.

```
CREATE TRIGGER Before_Update
BEFORE UPDATE ON Users
FOR EACH ROW
    INSERT INTO Users_Update_info
    SET action = 'UPDATE',
            User_id = OLD.U_id,
            User_Name = OLD.User_Name,
            Update_Date = NOW();
```

2. Trigger to check the Validity of Visitor's phone number

```
delimiter $$
CREATE TRIGGER Vistor_Entry
BEFORE INSERT ON Visitor
FOR EACH ROW
 BEGIN
  IF NEW.Visitor_Phone_no not rlike '^[0-9]{10}$'
   THEN
    signal sqlstates '45000' set message_text = 'INVALID PHONE NUMBER !!!' ;
  END IF;
END;
```

SQL Queries

1. **Retrieve Names of Societies which have complaint about 'Water problem'.**

```
SELECT Society_Name
FROM Society as S, House as H
WHERE S.S_id = H.Society_id AND H.H_id in (
                        SELECT House_no
                    FROM Users as U, Complain as C
                        WHERE U.U_id = C.User_id AND C.Subject = 'Water Problem');
```

2. **Retrieve Names and Phone numbers of all users who don't live in Society 'Home Palace' and have no complaints.**

```
SELECT User_Name, Phone_no
FROM Users
WHERE U_id not in (SELECT U1.U_id
                    FROM Users as U1, House as H
                    WHERE U1.House_no = H.H_id AND H.Society_id in (
                                SELECT S_id
                                        FROM Society
                                        WHERE Society_Name = 'Home
                                Palace' ))
                        AND U_id not in ( SELECT User_id
                                            FROM Complain );
```

OUTER JOIN Queries

3. **Retrieve all User names and their Phone numbers who have made atleast one Complain.**

```
SELECT Distinct User_Name, Phone_no
FROM Users LEFT OUTER JOIN Complain ON U_id = User_id
ORDER BY 1;
```

4. **Retrieve the Count of all the Users stored in the database living in each Society.**

```
SELECT S_id, Society_Name, COUNT(1) as No_of_Users
FROM (Users LEFT OUTER JOIN House ON House_no = H_id) LEFT OUTER JOIN
Society ON (S_id = Society_id)
GROUP BY S_id;
```

AGGREGATE FUNCTIONS :

1. **Retrieve Average, Maximum and Sum of Rent paid by all the residents of Society 'The Den'.**

```
SELECT avg(Rent_price), MAX(Rent_price), SUM(Rent_price)
FROM House as H
WHERE H.Society_id in (SELECT S_id
                       FROM Society
                            WHERE Society_Name = 'The Den') ;
```

2. **Retrieve Complaint subjects whose complaints are more than once.**

```
SELECT C.Subject,COUNT(*)
FROM Complain as C
GROUP BY C.Subject
HAVING COUNT(*) > 1;
```

3. **Find Names of all Rent payers whose rent is more than 30000.**

```
SELECT User_Name,Rent_price
FROM Users as U, House as H
WHERE U.House_no = H.H_id AND Rent_price >= 30000;
```

Conclusion

This system is very helpful for Societies with large number of residents, where the data is too big to be stored. This also helps for easy retrieval of information. The information retrieval also becomes easy as the way it is stored is easy to understand. This system can also support an environment where an user owns more than one house. And also the record of visits is kept separately which helps to save time and storage. This system can be extended to support joint ownerships and details about residents. Overall this system is enough for any Society to store enough data required for any situation.

FUTURE SCOPE :

- System can be extended to support more details about the owners or rent payers.
- The system can be extended to store details about each resident living in each house.
- This system can be paired with Java or PHP to make insertion or any operation easier.
- Can be extended to store information about facilities and their responsible persons information.
- The system can also be extended for parking lots of each society.
- This project can be enhanced further by Developing a Mobile App and a web platform and making it available to the end users.
- Also we can Develop a Full Fledged accounting module. The software is flexible enough to be modified and implemented as per future requirements.