DBMS MINI PROJECT

MOVIE TICKET RESERVATION MANAGEMENT SYSTEM

1. Problem Statement

Develop a database system that contains information about ticket reservations for movie performances. To make a reservation you must be registered as a user of the system. In order to register you choose a unique username and enter your name, address, and telephone number. When you use the system later, you just have to enter your username. In the system, a number of theatres show movies. Each theatre has a name and a number of seats. A movie is described by its name only. A movie may be shown several times, but then during different days. This means that each movie is shown at most once on any day. You can only reserve one ticket at a time to a performance and cannot reserve more tickets than are available at a performance. When you make a reservation, you receive a reservation number that you will use when you pick up the ticket.

Requirement analysis:

We are required to store the personal data of the user like username, phone number, contact address and password for login in 'login info' table. We need to store the details about the movies being screened and the details of various theatres available for ticket registration in two respective tables i.e. 'movies' and 'theatre'.

Finally, the information inserted by the user regarding the movie, theatre, date and time of the show are inserting into 'bookings' table. The contents of this table are used for reference purpose by the administration and also by the user to use it for convenience in referring to the registration id in it which is required for fetching the hard copy of the ticket from the theatre, in person, which is a pre-requisite.

Software requirement:

Graphical user interface – Python
Database creation and manipulation – SQL*Plus
Database – Oracle

Potential solution:

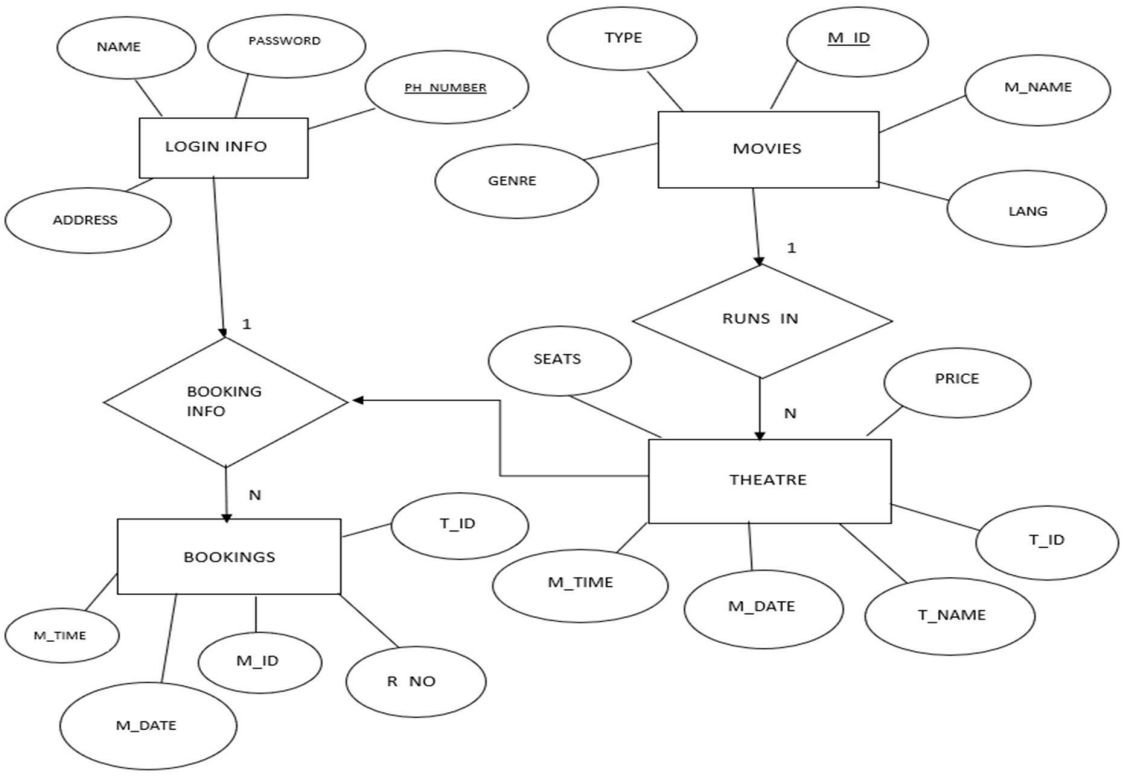
The objective of this project is to develop an efficient movie reservation management system. Initially, the user is provided with the option to either log into the reservation facility or to register himself as a user into the system if it's his/her first time using the management system. For registration purpose, the user is asked for a username, phone number, contact address and a password and it is stored in 'login info' table. The table is referred to authorize login attempts.

Once the user logs in into the system, options to view the movies currently being screened and the theatres in which the movies can be viewed are displayed. Details like genre and type of the movies are displayed to provide the user with a detailed insight of the movies. The theatre information contains information like the availability of seats, price of each ticket, date and time at which a particular movie is to be screened in the respective theatre.

Unique movie id and theatre id are displayed adjacent to the corresponding movies and theatres to make the system more user friendly. The users can type the movie id, theatre id, date and time preferred in the given spaces to reserve a ticket for the movie they want to watch in the theatre they prefer at a particular date and time.

The user is provided with a “Book ticket” button, which when pressed confirms the booking and reserves a ticket for the user. Finally, the ticket detail is displayed along with a unique reference number which can be used by the user for fetching hard copy of the ticket from the respective theatre. The user can then log out of the system or continue booking more tickets.

2. ER Diagram



3. Database Schema

LOGIN INFO TABLE:

Name	Phone_number	Address	Password

MOVIES TABLE:

M_ID	M_Name	Type	Language	Genre

THEATRE TABLE:

T_ID	M_ID	T_Name	M_Date	M_Time	Seats	Price

BOOKINGS TABLE:

Phone_Number	Reg_no	T_ID	M_ID	M_Date	M_Time

4. Database normalization

Normalization is a technique to reduce or remove redundancy from a table.
In this Normalization we have arrived with 4 tables with no redundancy for the database.
The process of arriving at the conclusion is discussed below:

1 NF:

Name	Password	P_number	Address	M_id	M_Name	Type	Language	Genre	T_id	T_Name	M_date	M_time	Seats	Price	R_no

Name	Password	P_number	Address

M_id	M_name	Type	Language	Genre	T_id	T_name	M_date	M_time	Seats	Availability	Price

Table should not contain multivalued attributes for a 1NF configuration. So, the table has been split up in such a way there are no multivalued attribute in the table.

2 NF:

P_number	T_id	M_id	M_time	M_date	R_no

T_id	M_id	M_name	T_Name	Type	Language	Genre	M_time	M_date	Seats	Price

In the 2nd normal form, the table must be in 1NF configuration and all the non-prime attributes should be fully functionally dependent on candidate key.

3 NF:

T_id	M_id	T_name	M_date	M_time	Seats	Price

T_id	T_name

M_id	M_Name	Type	Language	Genre

In the 3rd normal form, the table must be in 2NF and there should be no transitive dependency on the table.
So, you split the table further until you get no redundancy to get the tables for the required database.

Appendix-Report on Python GUI tkinter

Tkinter is the standard GUI library for Python. It is the Python interface to the Tk GUI toolkit shipped with Python.

To create a window in tkinter we user a Tk() object, which allows us to create a standalone window.

```
import Tkinter
top = Tkinter.Tk()
# Code to add widgets will go here...
top.mainloop()
```

The above code creates a blank window when executed.

Tkinter Widgets:

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

Brief description of all the Tkinter widgets used in this project.

1. Label:

Label widget implements a display box where you can place text or images. The text displayed by this widget can be updated at any time.

Syntax:

l = Label (master, option, ...)

Parameters:

- ***master*** – This represents the parent window.
- ***options*** – text, font, bg, image, justify, fg, bitmap etc.

2. Button:

The Button widget is used to add buttons in a Python application. These buttons can display text or images that convey the purpose of the buttons. You can attach a function or a method to a button which is called automatically when you click the button.

Syntax:

b = Button (master, option=value, ...)

Parameters:

- ***master*** – This represents the parent window.

- ***options*** **–** text, font, bg, fg, image, height, width, command etc.

3. Scrollbar:

This widget provides a slide controller that is used to implement vertical scrolled widgets, such as Listbox, Text and Canvas.

Syntax:

s = Scrollbar (master, option, ...)

Parameters:

- ***master*** **–** This represents the parent window.
- ***options*** **–** bg, bd, command, orient, width, repeatinterval, takefocus etc.

4. Toplevel:

Toplevel widgets work as windows that are directly managed by the window manager. They do not necessarily have a parent widget on top of them.

Syntax:

t = Toplevel (master, option, ...)

Parameters:

- ***master*** **–** This represents the parent window.
- ***options*** **–** bd, bg, cursor, height, width, relief, font, fg etc.