**Lewis University  
CPSC 50900: Database Systems  
Term Project**

# **Movie Theater Reservation System**

**Movie Theater Reservation System Design** is basically aimed to provide complete information of the movie and schedule to the customer, according to which he/she can easily get a ticket instantly and can book a ticket on his/her favorite movies.

Admin can use Movie Ticketing System to insert and delete data such as movie descriptions, movie schedules, seating and auditorium details which will update the related webpage and will be accessible by the customers.

Admin can update changing according to the data in the database also admin can check the statistics information from the system.

Operate your cinema with better efficiency by automating reservation and ticketing process improve profitability and manage your cinema better by having access to key data in a centralized and systematic view and increase customer satisfaction by giving your customers what they want when it comes to the seat preference.

**Anil Potru, anilpotru@lewisu.edu**

*https://github.com/*

Table of Contents

[Initial Proposal 2](#_Toc85814569)

[Data Sources 2](#_Toc85814570)

[Data Storage Alternatives 2](#_Toc85814571)

[Relational Database Design Process 3](#_Toc85814572)

[Relational Database Design 3](#_Toc85814573)

[Data Definition Language (DDL) Scripts 3](#_Toc85814574)

[Data Manipulation Language Scripts 4](#_Toc85814575)

[Indexes 5](#_Toc85814576)

[Views 5](#_Toc85814577)

[Triggers 5](#_Toc85814578)

[Transactions 5](#_Toc85814579)

[Database Security 6](#_Toc85814580)

[Locking and Concurrent Access 6](#_Toc85814581)

[Backing Up Your Database 6](#_Toc85814582)

[Python Programming 7](#_Toc85814583)

[PHP Programming 7](#_Toc85814584)

[Suggested Future Work 8](#_Toc85814585)

[Activity Log 8](#_Toc85814586)

# Initial Proposal

There are a few assumptions we have to bear in mind:

* contemporary multiplex movie theaters can have one or more auditoriums within a larger complex,
* each auditorium can have a different number of seats,
* seats are numerated with row number and seat position within a row,
* a movie can have multiple screenings at different times, or it can be screened simultaneously in a different auditorium,
* for each screening a seat can be reserved/sold only once,
* we want to track who entered each reservation/sale into the system.

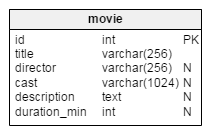
Admin can use Movie Ticketing System to insert and delete data such as movie descriptions, movie schedules, seating and auditorium details which will update the related webpage and will be accessible by the customers.

Admin can update changing according to the data in the database also admin can check the statistics information from the system.

Operate your cinema with better efficiency by automating reservation and ticketing process improve profitability and manage your cinema better by having access to key data in a centralized and systematic view and increase customer satisfaction by giving your customers what they want when it comes to the seat preference.

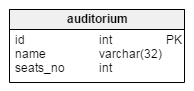
**Short table structure descriptions are given below:**

1. The movie table contains data about movies which will be shown in the theater. The primary key is id, which is auto\_incremented like all primary keys in all other tables. The only mandatory data is title.



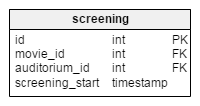
All fields have meanings according to their name. The column duration\_min could be used to disable inserting a new screening or to show an alert message in case we want to enter a screening in an auditorium where the previous screening is still in progress:  
previous screening start time + duration\_min of it > this screening start time

1. The auditorium table identifies all auditoriums in theater. All data is mandatory.



The seats\_no field can be used to calculate percentage of availability of auditoriums for a selected screening/movie/auditorium/date range. This is an example of data redundancy because we could get the number of seats for each auditorium by counting them in the seat table. In this example it might not improve performance significantly. I show it here as an idea that could help with designing more complex models. If we set up the database this way we must bear in mind that if we change one piece of data, we also have to change others. If we add or delete data from the seat table we have to adjust values seats\_no in the auditorium table.

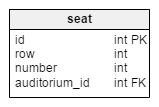
1. The screening table contains data of all screenings and all fields are mandatory. A screening must have a related movie, auditorium and start time. We can’t have two showings in same auditorium at the same time. We can define a unique key consisting of auditorium\_id and screening\_start. This setup is better than defining a unique key consisting of movie\_id, auditorium\_id, and screening\_start because that would allow us to enter screenings of two different movies at the same time in the same auditorium.



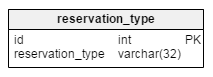
Vertabelo SQL preview code for this table looks like this

|  |
| --- |
|  |
|  |

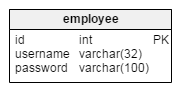
1. The seat table contains a list of all seats we have in auditoriums with each seat assigned to strictly one auditorium. All fields are mandatory.



1. The reservation\_type table is a dictionary of all reservation types (by phone, online, in person). All fields are mandatory.

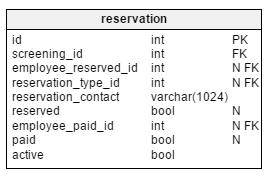


1. The employee table lists all employees using the system. All fields are mandatory.

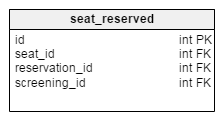


In complex systems there are usually more roles so we need to have a role dictionary and employee/user-role connection. In our example we have only one role: the same person inserts reservations and sells tickets.

1. The reservation and seat\_reserved tables are the main tables of our system. This is why I listed them last. All other tables can exist without reservation tables but without the reservation tables we would lose the reason for designing the whole database in the first place.



The reservation table stores data about a ticket reservation and/or sale. If we have a reservation, the attribute reserved would be set to True, the reservation\_type\_id would be set according to the origin of the reservation and the employee\_reserved\_id would contain the id\_employee value of the person who entered data (it would be empty if the reservation had been done online by the customer). In the same way, if tickets were sold, the employee\_paid\_id would be filled with the id\_employee value of the person who sold tickets, the attribute paid would be set to True. The active attribute identifies if a record is still valid. If tickets were sold this attribute would always be True and the reservation without sales would be active until 30 min before screening starts



The seat\_reserved table enables us to make a reservation or one payment for multiple seats. After the employee checks a few free seats on the interface, one record would be added to this table for each of them. If we want to check which seats are free or taken we can check the values in this table joined to the reservation table where reservation.active = True.

Additionally an admin can update the **webpage** changing the movies, schedules, auditoriums information according to the database. And also can use the **webpage** to check the statistics information from the system.

# Data Sources

**Movie Theater Reservation System Design** entities and their attributes:

**Movie Entity**: Attributes of Movie are id, title, director, cast, description, duration\_min

**Auditorium Entity**: Attributes of Auditorium are id, name, seats\_no

**Screening Entity**: Attributes of Screening are id, movie\_id, auditorium,id, screening\_start

**Seat**: Attributes of Seat are id, row, number, auditorium\_id

**Reservation\_type**: Attributes of Reservation\_type are id, reservation\_type

**Employee**: Attributes of Employee are id, username, password

**Reservation**: Attributes of Reservation are id, screening\_id, employee\_reserve\_id, reservation\_type\_id, reservation\_contact, reserved, employee\_paid\_id, paid, active

**Seat\_reserved**: Attributes of Seat\_reserved are id, seat\_id, reservation\_id, screening\_id

Please find the github link for the sample data sources attachments required for the above entities.

Below is the sample data of the movie entity json

# 

We have seen that RDBMS is better approach for the above database design. Where it guarantees the granularity and relationship between the tables.

However, as the system grows if we have below requirements the existing RDBMS may not be suitable for future requirements.

1. We have tons of data like movies information, actors, crew, comments and reviews

2. We need to have faster search

3. We need to run queries for analytics

So, alternatively we can use NoSQL as the option to acknowledge the above issues. MongoDB, Cassandra, and CouchDB are the examples of NoSQL.

# Relational Database Design Process

*Description: Consider the list of fields you identified in part c. Identify functional dependencies that exist among them. For each functional dependency, identify the determinants and the fields they determine. This becomes the basis for identifying your entity sets, which then become your tables. Give each entity set or table you identify in this way a unique and clear name, making sure that the names you use are singular nouns. Then list the relationships that exist among the various entity sets. For each relationship, identify its connectivity (one-to-one, one-to-many, many-to-many) and participation (optional or mandatory). Finally, make sure that none of the attributes you've assigned to each entity set are multi-valued. If they are, take the steps needed to break them down.*

*Rubric: Your work will be graded as follows:*

* *8 points for identifying all the functional dependencies, including determinants and the columns whose values they determine.*
* *2 points for naming the entity sets that make up your data with clear, easy-to-understand names.*
* *6 points for identify the relationships among the entity sets and identifying connectivity and participation for each.*
* *2 points for breaking down multi-valued attributes.*

*Total points possible: 18*

ENTER YOUR RELATIONAL DATABASE DESIGN DESCRIPTION HERE. INCLUDE SOURCE CODE AND SCREEN SHOTS.

# Relational Database Design

*Description: This is where you will complete your database design. For each of the entity sets you identified in the preceding section, analyze them to make sure they pass 2nd, 3rd, 4th, and Boyce-Codd Normal Form. If they do not, introduce additional entity sets or key changes to make sure that they do. Then, add foreign keys to connect entity sets that are related. For many-to-many relationships, introduce bridge entity sets to convert them into two one-to-many relationships. Also, consider whether you should introduce surrogate keys to create a more efficient primary key for some of your entity sets. Finally, diagram your design in Vertabello. Make sure your ER diagram correctly shows all entity sets, their primary and foreign keys, the data types for each attribute, and the connectivity and participation characteristics of each entity set. Your final Vertabello design should be something you could actually implement in a relational database management system.*

*Rubric: Your work will be graded as follows:*

* *4 points for the normalization analysis of your entity sets.*
* *3 points for introducing bridge entity sets.*
* *3 points for choosing foreign keys and perhaps more efficient surrogate keys*
* *10 points for correctly depicting your physical database model in Vertabello*

*Total points possible: 20*

ENTER YOUR RELATIONAL DATABASE DESIGN HERE

# Data Definition Language (DDL) Scripts

*Description: Use Vertabello to generate a script of SQL commands that build the database and its table structures. Write scripts or build Excel spreadsheets that take your data files and generate scripts of SQL insert statements from them. Use the MySQL source command to run the various scripts needed to build and populate the database in MySQL. Include the source code and / or Excel spreadsheets you use to manipulate and populate the data. Make sure all your tables have at least three records in them and that you've linked the tables through their foreign keys.*

*Rubric: Your work will be grades as follows:*

* *Database and table creation statements from Vertabello saved as an sql script file: 3 points*
* *Scripts you write or Excel spreadsheets you create to generate SQL commands for populating the tables, uploaded to GitHub: 8 points*
* *Descriptions of the scripts and Excel spreadsheets you wrote along with code excerpts included in the Word document: 5 points*
* *Screenshots of your successful attempts to use the MySQL source command to populate each table with at least three records: 4 points*

*Total points possible: 20*

ENTER YOUR DDL WORK HERE

# Data Manipulation Language Scripts

*Description: Write the SQL commands for twelve queries. Two queries should be insert statements, two should update statements, one should be a delete statement, one should be a simple select statement that selects a subset of the rows and columns from one table, two should be a select statements that select data from a joining of two tables, two should use summary functions to generate statistics about the data, one should be a multi-table query, and one should be another query of your choice. Show the queries and screenshots of the results in your Word document, and save your queries in a commented sql script to GitHub.*

*Rubric: Your work will be graded as follows:*

* *1 point each for the two insert statements*
* *1 point each for the two update statements*
* *1 point for the delete statement*
* *1 point for the simple select statement*
* *2 points each for the 2 join statements*
* *2 points each for the two that use summary statements*
* *2 points for the multi-table query*
* *2 points for the query of your choice.*
* *12 points for showing the query and a screenshot of the corresponding result set back-to-back for each of these queries in your Word document.*

*Total points possible: 30*

ENTER DML WORK HERE

# Indexes

*Description: Improve the performance of your design by adding indexes to various tables. Show the SQL needed to add the indexes. Explain why you chose the ones you added. Explain how you would demonstrate the impact the indexes had on the performance of various queries.*

*Rubric: Your work will be graded as follows:*

* *6 points for clearly defining at least three indexes and explaining why you chose them.*
* *3 points for showing the sql needed to generate the indexes*
* *3 points for explaining how you would demonstrate the performance improvement afforded by the indexes.*

*Total points possible: 12*

ENTER YOUR INDEX WORK HERE

# Views

*Description: Add two views to your database to provide easy access to combinations of data from multiple tables.*

*Rubric: Your work will be graded as follows:*

* *2 points for including the SQL for generating the two views in your Word document*
* *2 points for including screenshots for the data contained in each view in your Word document*
* *2 points for explaining why each view is a valuable addition to your database*
* *2 points for explaining who might benefit most from having access to each view.*

*Total points possible: 8*

ENTER YOUR WORK WITH VIEWS HERE

# Triggers

*Description: Add a trigger to a table so that data will be updated when a certain event occurs*

*Rubric: Your work will be graded as follows:*

* *2 points for including the SQL for the trigger in your Word document*
* *2 points for clearly explaining the purpose of the trigger*
* *2 points for a screenshot and explanation that shows the trigger in action.*

*Total points possible: 6*

ENTER YOUR WORK WITH TRIGGERS HERE

# Transactions

*Description: Demonstrate that you know how to define and use a transaction. Why are transactions important for ensuring ACID behavior?*

*Rubric: Your work will be graded as follows:*

* *3 points for clearly explaining the importance of transactions to ensuring ACID behavior*
* *3 points for including a screenshot and accompanying explanation of a MySQL transaction.*

*Total points possible: 6*

ENTER YOUR WORK WITH TRANSACTIONS HERE

# Database Security

*Description: Identify the different kinds of users who will use your database. Write GRANT statements to define the privileges for these different kinds of users.*

*Rubric: Your work will be graded as follows:*

* *6 points for clearly identifying and describing the various kinds of users who will use the databases and identifying and justifying what privileges each should have.*
* *4 points for writing GRANT statements that assign privileges to these different kinds of users.*
* *4 points for demonstrating with screenshots that your GRANT statements do distinguish among different kinds of users in regard to what they can do with the database.*

*Total points possible: 14*

ENTER YOUR WORK WITH DATABASE SECURITY HERE

# Locking and Concurrent Access

*Description: Explain the purpose of locking tables and show how to do that to prevent inconsistencies that may arise in your data when concurrent transactions take place.*

*Rubric: Your work will be graded as follows:*

* *3 points for clearly explaining an example that shows why you should lock tables to prevent inconsistencies.*
* *3 points for providing a screenshot and accompanying explanation of locking tables.*

*Total points possible: 6*

ENTER YOUR WORK WITH LOCKING AND CONCURRENT ACCESS HERE

# Backing Up Your Database

*Description: How you will back up your database. What commands will you issue? How frequently will the commands run? How can they be automated? Where will the backups be stored?*

*Rubric: Your work will be graded as follows:*

* *12 points for clearly explaining and justifying your database backup strategy, including the frequency with which you will back up the database, how you will automate backups, where you will store them, and how you will secure them. You will earn three points for addressing each factor (frequency, location, automation, and security)*
* *3 points for providing a screenshot of the command you would issue to back up the database and for including a portion of the resulting file.*

*Total points possible: 15*

ENTER YOUR WORK ON DATABASE BACKUPS HERE

# Python Programming

*Description: Write a Python program that generates a report that contains a subset of the data from your database. Include the code for your Python program in your Word document, and also post the program to your GitHub repository.*

*Rubric: Your work will be graded as follows:*

* *12 points for writing a Python script (and including its code in the Word doc) that will pull data from a database and store it to a text file and present it to the screen. Your code must have comments in it that explain how it works. You will be awarded 3 points for successfully connecting to the database, 3 points for successfully querying it, and 4 points for presenting the data to the screen and to a file. Internal comments count for 2 points.*
* *2 points for posting the code to GitHub*
* *4 points for showing a screenshot of your running the script and showing the results it produces on the screen.*

*Total points possible: 18*

ENTER YOUR PYTHON DATABASE PROGRAMMING WORK HERE

# PHP Programming

*Description: Build an HTML form that enables the user to specify criteria to search by. Use PHP to show the results of the query on a resulting web page. Make sure you include protections against an SQL injection attack. Include your HTML and PHP code in your Word document, and also post the files to your GitHub repository.*

*Rubric: Your work will be graded as follows:*

* *4 points for writing an HTML form the user will use to enter search criteria*
* *8 points for a PHP script that uses the search criteria and returns results*
* *4 points for an HTML page that shows the results*
* *4 points for explaining what SQL injection might be run on your website and explaining how you prevented it.*
* *4 points for providing screen shots of your PHP website in action.*
* *2 points for posting your code to GitHub*

*Total points possible: 26*

ENTER YOUR PHP DATABASE APP PROGRAMMING WORK HERE

# Suggested Future Work

*Description: Describe the limitations of your current database and explain how you or someone else could improve the design to address these shortcomings. Also describe how you might take advantage of leverage cloud services to increase the performance and availability of your database. Finally, explain the advantages and disadvantages of storing your data in a NoSQL format instead.*

*Rubric: Your work will be graded as follows:*

* *3 points for clearly describing the limitations of your databases*
* *3 points for explaining how you would address these shortcomings*
* *3 points for explaining how you might migrate the database to the cloud and describing what advantages you might gain from doing that.*
* *3 points for explaining the advantages and disadvantages of storing your data in a document-based NoSQL format instead.*

*Total points possible: 12*

ENTER YOUR SUGGESTED FUTURE WORK IDEAS HERE

# Activity Log

*Description: As an appendix, the team will keep a daily diary or log of their activity. What did you or your team study in this class each day? What did you learn? What did you accomplish or build or design? You don't have to enter something every day, but there should be at least three entries each week. Since we have eight weeks, that means you should make 3 posts to the Activity Log each week, for a total of at least 24 posts. Each post will be worth 1 point.*

*If you are working as part of a team, make sure you clearly identify which team member worked on which tasks. The Activity Log should help me figure out how each team member contributed to the project. If I cannot discern who worked on what aspects of the project from the activity log, no points will be awarded for it.*

*Total points possible: 24*

MAKE AT LEAST THREE ENTRIES PER WEEK. CLEARLY IDENTIFY WHAT EACH PERSON ON YOUR TEAM ACCOMPLISHED. YOU MUST SHARE THE RESPONSIBILITY OF COMPLETING THE PROJECT.