**Database Model**

**Cinema Movie Ticket Booking System**

Ngo Trong Hieu (15943) – Software Engineering

***Note:*** *this report with all the relevant resources could be accessed on GitHub:*

[*https://github.com/cpulover-university/cinema-movie-ticket-booking-system-database-model*](https://github.com/cpulover-university/cinema-movie-ticket-booking-system-database-model)

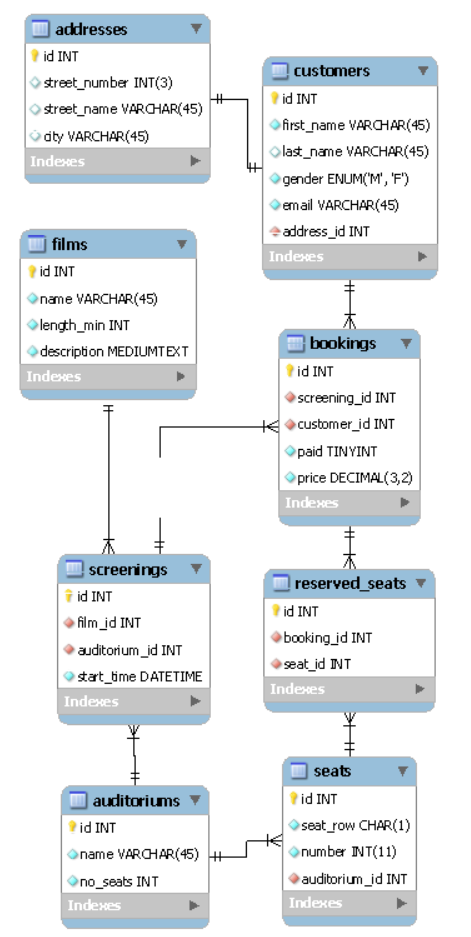
# Requirements

Personally (individual work) create a relational database system (model and implementation) regarding specific subject (e-commerce system, system for handling personnel and activities of certain organization, system for ordering and registering goods, books, movies, songs repository, etc.).

You can use MySQL database with phpMyAdmin panel or other tools. Task includes:

1. Modelling of a system (creation of relational diagrams, ERD diagrams, establishing keys and providing normalization).
2. Creating tables and filling them with data (records).
3. Being familiar with data types and providing basic SQL queries.
4. Providing more advanced SQL queries (queries to multiple tables, aggregated queries and grouping).
5. Providing nested queries.

# Relational/ERD diagram [1]



# Descriptions

The ***films*** table contains data about films which will be shown in the cinema. The only mandatory data is title.

The ***auditorium*** table identifies all auditoriums in the cinema. All data is mandatory.

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.

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

The ***customers*** table lists all customers using the system. All fields are mandatory.

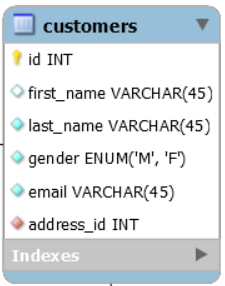
The ***addresses*** table.

The ***bookings*** table stores data about all of ticket reservations. The customer\_id would contain id of the customer who books ticket. In the same way, if tickets were sold, the attribute paid would be set to True.

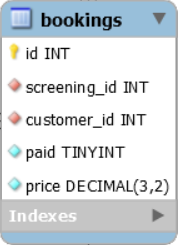
The **reserved\_seats** table enables us to make a reservation or one payment for multiple seats. After the employee checks a few free seats on the interface, one record would be added to this table for each of them.

# Key establishments [1]

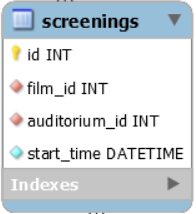
In all tables, ***id*** is the primary key, which is auto incremented.



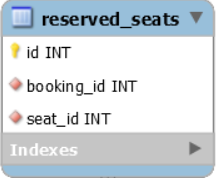
In customers table, ***address\_id*** is the foreign key referencing to id of address table. The relationship is one to one: one customer has one address.

**

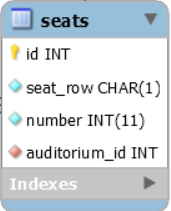
In bookings table, ***screening\_id*** foreign key and ***customer\_id*** references to the id in screening table and customer table, respectively. Therefore, one customer and one screening form a unique booking.

**

In screenings table, ***film\_id*** foreign key and ***auditorium\_id*** references to the id in films table and auditoriums table, respectively. Therefore, one film and one auditorium form a unique screening.



In reserved\_seats table, ***seat\_id*** foreign key and ***booking\_id*** references to the id in seats table and bookings table, respectively. Therefore, one seat and one booking form a unique reserved seat.



In seats table, ***auditorium\_id*** is the foreign key referencing to id of auritoriums table. The relationship is many to one: many seats may belong to one auditorium.

# Normalization [1]

## 1NF identification

In all tables, it could be noticed that:

* All the columns have unique names in each table.
* Each column contains atomic values.
* Combination of data is unique for each row.
* The database is in 1NF.

## 2NF identification

Next, in all tables, there is no partial dependency: all non-key attributes are dependent on the whole set of key attributes (primary key).

* The database is in 2NF.

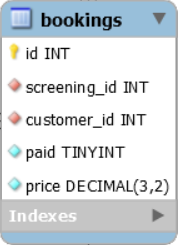
## 3NF identification

Next, in all tables, there is no transitive dependency: all non-key attributes are not dependent on other non-key attributes in the same table.

* The database is in 3NF.

## BCNF identification

In bookings table, screening\_id and customer\_id forms a candidate key:

**

However, there is also a functional dependency between price and screening\_id, while screening\_id alone is not a primary key.

* This table is not in BCNF.

*In conclusion, all the tables in database is in 3NF.*

# Used data types [3]

* **INT:** id. A normal-sized integer that can be signed or unsigned. If signed, the allowable range is from -2147483648 to 2147483647.
* **DECIMAL(3, 2):** price (bookings table)
* **VARCHAR:** first\_name (customers table). A variable-length string between 1 and 255 characters in length.
* **CHAR(1):** seat\_row (seats table). A fixed-length string between 1 and 255 characters in length, in this case is 1.
* **BOOLEAN/TINYINT(1):** paid (bookings table)
* **ENUM(‘M’,’F):** gender (customers table). Define columns that can contain only a given set of values.
* **DATETIME:** start\_time (screenings table). A date and time combination in YYYY-MM-DD HH:MM:SS format, between 1000-01-01 00:00:00 and 9999-12-31 23:59:59.
* **MEDIUMTEXT:** description (films table). A field with a maximum length of 16777215 characters.

# Create & populate database [2]

* Create database: Forward Engineer the EER diagram in MySQL Workbench.
* Populate data: [Generatedata](https://www.generatedata.com/) website
* Format query: [Dpriver](http://www.dpriver.com/pp/sqlformat.htm) website
* Generate 11 customers, 10 addresses, 10 films, 4 auditoriums, 20 screenings, 100 seats, 100 bookings.

All scripts are attached along with this report.

# Basic queries [3]

Attached script: basic-queries.sql

Select, as, and, or, order, in, where, limit, between, concat.

# Advanced queries [4]

## Attached script: advanced-queries.sql

## Queries to multiple tables

Inner join, left/right join.

## Aggregate queries & grouping

Count, count distinct, sum, min/max, average, grouping, having.

# Nested queries [5]

## Non-correlated queries

## Correlated queries