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

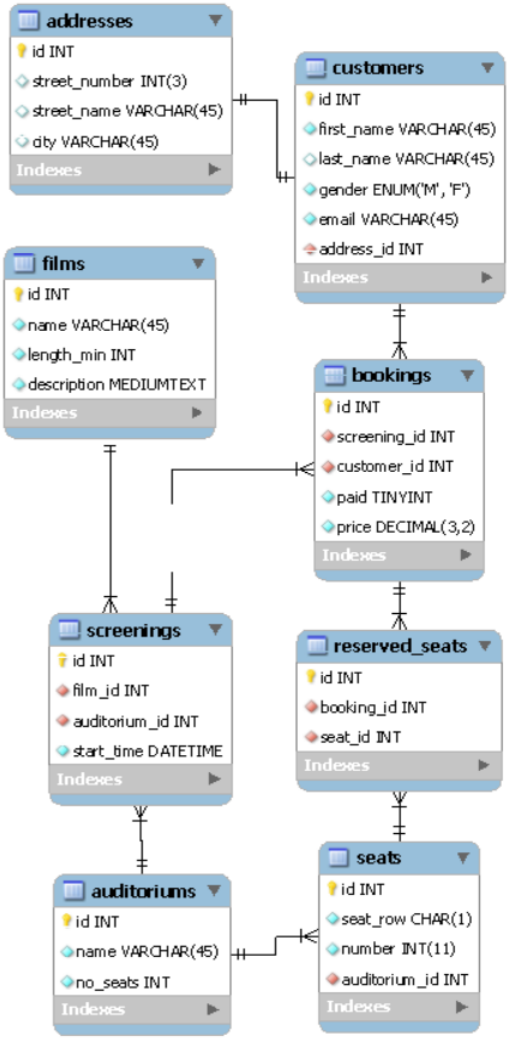
REQUIREMENTS

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 **customers** table stores customers booking tickets. Only last name and address id fields are optional.

The ***addresses*** table stores addresses of customers with each one assigned to one customer. Only id field is mandatory.

The ***films*** table stores films which are shown in the cinema. The length_min is length of a film in minute. All fields are mandatory.

The ***auditorium*** table stores data of all auditoriums in the cinema. The no_seats is number of seats in an auditorium. All fields are mandatory.

The ***screening*** table stores data of all screenings which are identified by each unique pair of a film and an auditorium. All fields are mandatory.

The ***seats*** table stores all the seats in auditoriums with each seat assigned to strictly one auditorium. All fields are mandatory.

The ***bookings*** table stores all the ticket reservations with each one assigned to one customer and one screening. If tickets were sold, the attribute paid would be set to 1 (true). All fields are mandatory.

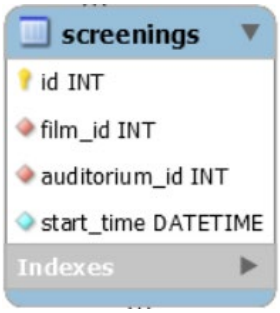
The ***reserved_seats*** table stores all the seats which are reserved by booking. Therefore, each one is assigned to one booking and one seat. All fields are mandatory.

KEY ESTABLISHMENTS [1]

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

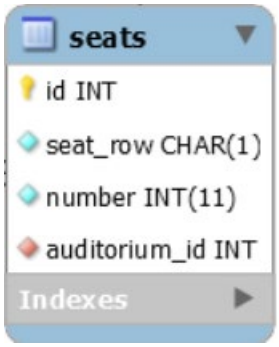
In customers table, ***address_id*** is the foreign key referencing to id of address table. The relationship is one to one: one customer has one address.

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.

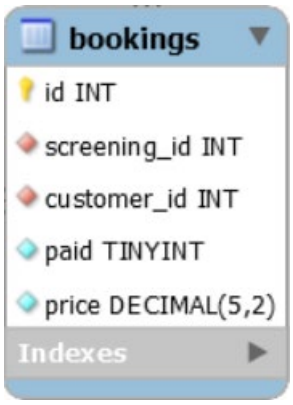
3NF identification

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 are in 3NF.

USED DATA TYPES [3]

- **INT:** id in all tables - a normal-sized integer with the allowable range from 0 to 4294967295 if unsigned.
- **DECIMAL(5, 2):** price in bookings table - an unpacked floating-point number storing 5 digits with 2 decimals.
- **VARCHAR:** first_name in customers table a variable-length string between 1 and 255 characters in length.
- **CHAR(1):** seat_row in seats table - a fixed-length string between 1 and 255 characters in length, in this case is 1.
- **BOOLEAN - TINYINT(1):** paid in bookings table, could be set to 1 (True) or 0 (False).
- **ENUM('M', 'F):** gender in customers table - which contains only a value in the given set: M (Male) or F (Female).
- **DATETIME:** start_time in screenings table – a date and time combination in YYYY-MM-DD HH:MM:SS format.
- **MEDIUMTEXT:** description in films table - a string 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 website
- Format query: Dpriver website

↦ Generate 11 customers, 10 addresses, 10 films, 4 auditoriums, 20 screenings, 100 seats and 100 bookings.

📎 All scripts are attached along with this report.

BASIC QUERIES [3]

📎 *Attached script: basic-queries.sql*

Modifying data queries

INSERT, UPDATE, DELETE, INSERT IGNORE.

Selecting data queries

SELECT, AS, AND, OR, ORDER, IN, WHERE, NULL, LIMIT, BETWEEN, CONCAT.

ADVANCED QUERIES [4]

📎 *Attached script: advanced-queries.sql*

Queries to multiple tables

INNER JOIN, LEFT/RIGHT JOIN, CROSS JOIN.

Aggregate queries & grouping

COUNT, COUNT DISTINCT, SUM, MIN, MAX, AVERAGE, GROUPING, HAVING.

NESTED QUERIES [5]

📎 *Attached script: nested-queries.sql*

Non-correlated query, non-correlated query with a derived table, correlated query.