Theater Reservation System

Group 24

Aditi Sonik (1211009433), Bavithra Ramakrishna (1210888611), Pon Arasu Neranj Subramanian (1209360889), Sarthak Khanna (1211255640)

Project video **:** https://www.youtube.com/watch?v=ASg02X8B7UQ&feature=youtu.be

A theater reservation system, where an administrator can keep track of the halls available in its theater, the shows that are taking place in the theater, reservations made by the shows for the halls, events that are scheduled for those shows and interested customers placing booking to attend the shows.
The halls in the theater are of two types namely auditoriums and screens.The shows are also of two types performances and movies.The performance takes place at the auditoriums and the movies are screened at the screens.
In our reservation system, a show can book a hall multiple times but each hall should be reserved only by one show for a particular time. Similarly a show can have a number of events but each event should be associated with only one show.
The customers who are interested to see the shows that are taking place at the theater will book tickets for the event of the show that they want to watch. Thus each booking is associated with a customer and an event. A customer can place multiple bookings but he has to place a booking to become a customer. Each booking will only have one customer and one event.

The administrator can query the database and get to know the following details.
The various categories of the performance that are taking place at the theater their counts, the different movies that are being screened and the number of movies that were released each year. The number of reservations that each show has made and also the number of reservations that has been made on each hall. The number of events that each show has. The shows that fall under a particular category of performance. The movies that come under a particular rating. The movies or the performances that are taking place at a particular hall. The number of bookings that each customer has placed and also the customer who has booked the maximum number of tickets on a particular booking. The top ten customers who has done the maximum number of bookings. The ticket price of the shows in the theater and the least popular shows at the theater can also be tracked. The above mentioned queries can be extended to finding the total number of auditorium or screen, which show are running at which halls, the time and date of reservations of a show and so on.

And a user can make a booking for a particular instance of a show that is being held at the theater.

The dummy data used for our model has some assumptions.

- Reservations for the halls by show organizers are carried from August 2016 to December 2016.
- Events of the shows are scheduled to be aired from August 2017 to December 2017.
- Booking window for interested customers is opened from January 2017 to July 2017.
- An event for a show can be on any of the seven days in a week.
- Every day has fixed time slots, assuming each event to be maximum of three-hour duration.
- Time slots for each day are 9:00am, 12:00pm, 3:00pm, 6:00pm, 9:00pm.
- We have taken event tickets to be one among the price bucket of 20, 25, 30, 35, 40, 45, 50.
- Performances performed are among 'Play', 'Drama', 'Opera', 'Dance', 'Magic Show'.
- Movies consists of rating among 'R', 'PG-13', 'A', 'U', 'PG-16'.
- Movie screens can be of type 'Dome', 'Flat', 'Curve'.
- Screen experience can be '2D', '3D', or '4D'.
- Number of ‘Green rooms’ within an auditorium range between 1 to 10.
- Customer can book tickets within range 1 to 10.
- Auditorium and screen sizes ranges from 4000 sq feets to 9999 sq feets.
- Each hall has a capacity of accommodating 100 people.
- Our model has 1000 halls out of which first 500 are screens and next 500 are auditoriums.
- It consists of 1000 shows out of which first 500 are movies and next 500 are performances.
- We assume 2000 reservations already placed for the halls.
- We are taking at least 2000 events scheduled for reserved shows.
- We are assuming 1000 customers have already placed their booking for the events, making customer table of 1000 entries.
- Booking table consisting of customer bookings already consist of 3000 booking.

Entities

Hall - The building complex (Theater) contains various rooms, called halls. The hall are of 2 types, auditorium or screen.
A hall is either a screen or an auditorium;Covering constraint.
Attributes- **Hall_id**, hall_name, capacity, availability_label, location.

Auditorium – Subclass of hall, where performances can be performed.
Attributes- Stage_size, Number of green rooms.

Screen- Subclass of hall, where movies can be screened.
Attributes- Screen_size, screen_type and screen_experience(2D, 3D, 4D).

Show- The show is an either a performance or a movie that is taking place in the theater complex.
The show should either be a movie or a performance;Covering Constraint.
Attributes- **Show_id**, Show_name.

Movie- Subclass of show.
Attributes- Release_date, cast, director, genre.

Performance- Subclass of show.
Attributes- Performance_type, performers.

Reservation - For a show to take place at a particular hall, the hall has to be reserved for a time slot. The record of a reservation made by a show for a hall is modeled by this entity. Every show should have a reservation for a show to take place. Each reservation is associated with only one show and one hall. But a show can have multiple reservations.
Attributes- **r_id**, r_date r_price.

Event - It is a particular instance of a show at a given date, time and place. A show can have multiple events but each event will be associated to a single show. It is denoted by eventTable attribute.
Attributes- **Show_id**, event_id, event_time, event_date, ticket_price.

Customer - A person who wants to watch the show at the theater and he/she makes a booking for the show.
Attributes- **Customer_id**, name, phone_number, email_id, payment_details.

Booking - Customer makes a booking for an event, i.e, purchases a ticket and reserves a seat.Booking entity captures an instance of this.
An event can have many bookings but a booking is associated to only one event and one customer.
Attributes- **booking_id**, booking_time, booking_date, number_of_tickets, price, booking_label.

Queries

Administrator
The queries are from the perspective of the admin. The user inputs over here are the inputs that are given by the admin to the database.

1. Admin can add events to shows (Query with insert)

We know that, a show can have many events and events are nothing but instances of shows. So in our database the admin has the access to add an event to any show

```
SET @show_id='1';

SET @time_event = "21:00:00";

set @date_event = "2017-04-03";

set @ticket_price = "12";

insert into eventtable(show_id,time_event,date_event,ticket_price)
values(@show_id,@time_event,@date_event,@ticket_price);
```

This is a simple operation where the admin inserts the show_id, time_event, date_event, ticket_price which are the attributes of the event table to the show. The event id is auto incremented when a new event is added.

2. Admin can view the Number of reservations for selected show (Select query with groupby)

For a selected show the admin can view the number of events that has been created for it.

```
USER INPUT : Show_name

SET @show_name  = '%Liumouth%';
SELECT
show_name, COUNT(reservation.show_id) AS num_of_reservations
FROM
shows
JOIN
reservation ON reservation.show_id = shows.show_id
WHERE
show_name like @show_name
GROUP BY reservation.show_id
```

3. The admin can view the top customers who has made the maximum number of bookings. (Select query with group_by and aggregate)

The admin can select the number of customers that he wants to view.So the customers who has booked the maximum number of tickets are displayed.By this query the admin can get to know the top customers who has made many booking and can give offers to them.

```
SELECT
    customer_name, COUNT(booking.customer_id)
FROM
    booking,
    customer
WHERE
    booking.customer_id = customer.customer_id
GROUP BY booking.customer_id
ORDER BY COUNT(booking.booking_id) DESC
LIMIT 10
```

4.**Number of movies released in the year selected by the admin**

For the selected year the admin can view the number of movies that was screened at the theatre.

```
USER INPUT = release_year

set @release_date = '2000';

SELECT
release_date, COUNT(shows.show_id) AS Total_movie
FROM
shows
JOIN
movie ON movie.show_id = shows.show_id
WHERE
release_date = @release_date
GROUP BY release_date
```

5.**The admin can view the e_id and the ticket price for a selected show**

For a selected show_name the admin can view all the event ids and the corresponding ticket prices.(Correlated query)

```
User input: Show_name

SET @show_name = '%Rodriguezshire%';

SELECT
e_id, ticket_price
FROM
eventtable
WHERE
eventtable.show_id in (SELECT
shows.show_id
FROM
```

```
shows
WHERE
show_name like @show_name)
```

6. The admin can view the show_name and the performers for a selected performance type

For a selected type of performance ,the admin can view all the show_names that fall in that particular type of performance.

```
User input: performance_type

SET @performance_type  = '%Opera%';

SELECT
show_name
FROM
performance,
shows
WHERE
performance_type like @performance_type
AND shows.show_id = performance.show_id
```

7. Admin can view the number of bookings for each event

For a selected event the admin can view how many tickets has been booked for that particular event.

```
User input : event_id

Set @e_id = '1809 ' ;

SELECT
COUNT(booking_id)
FROM
booking
WHERE
e_id = @e_id
GROUP BY e_id
```

8. Admin can view the names of the customers who has booked the selected number of tickets.

The admin can view the customers who has booked the selected number of tickets.
Eg., If admin wants to see the customers who has booked 10 tickets or so to give them a discount on their next ticket.

```
User_input : num_tickets

SET @maz_tickets = '2';
```

```
SELECT
customer_name
FROM
(SELECT
customer_id, COUNT(booking.num_tickets)
FROM
booking
GROUP BY booking.customer_id
HAVING COUNT(booking.num_tickets) = @maz_tickets) AS s,
customer
WHERE
s.customer_id = customer.customer_id
```

9. The halls at which the particular event is taking place

The admin can see the hall at which the particular event is taking place given an event_id as input.

```
User input: event_id

set @e_id = '1';
SELECT
hall.name_hall
FROM
hall
WHERE
hall.hall_id IN (SELECT
reservation.hall_id
FROM
reservation
WHERE
reservation.show_id IN (SELECT
show_id
FROM
eventtable
WHERE
e_id = @e_id))
```

10. The date and time of reservations and the show_name for a particular show_id

For a show to take place a reservation has to be made. So for a selected show_id the admin can view the time and the date if the reservations made for that particular show.

```
User input :  show_id

SET @show_id='100';
```

```
SELECT
show_name, reservation.r_date, reservation.r_time
FROM
reservation
JOIN
shows ON reservation.show_id = shows.show_id
WHERE
reservation.show_id = @show_id
```

11. The details of the customer who has placed a particular booking

For every booking is placed by a customer. So given the booking id, the admin can view the customer details for that selected booking_id.

```
User input : Booking_id

set @booking_id = '10';
SELECT
*
FROM
customer
HAVING customer_id IN (SELECT
booking.customer_id
FROM
booking
WHERE
booking_id = @booking_id)
```

12. The shows that are screened at a particular hall for a given hall_id.

For a selected hall_id, the admin can view all the shows that are taking place at that particular hall.

```
User input : hall_id

set @reservation.hall_id = '319';

SELECT
shows.show_name
FROM
shows
WHERE
shows.show_id IN (SELECT
eventtable.show_id
FROM
eventtable
HAVING eventtable.show_id IN (SELECT
```

```
reservation.show_id
FROM
reservation
WHERE
reservation.hall_id = @reservation.hall_id))
```

13. The films with a particular rating(R,PG-13,PG-16,A,U)

For a selected rating, the admin can view the names of the movies that fall under that particular rating.

```
User input : rating

SET @rating = '%R%';
SELECT
shows.show_name
FROM
shows
WHERE
shows.show_id IN (SELECT ALL
movie.show_id
FROM
movie
WHERE
rating like @rating
GROUP BY movie.rating)
```

14. The shows having greater than the selected number of events

The admin can view the show name and the release date of the shows that have more than the selected number of events. This can help the admin to get to know the the popular shows.

```
User_inpt : A number

set @cnum=6;
SELECT DISTINCT
show_name, release_date
FROM
movie, shows
WHERE
shows.show_id IN (SELECT
show_id
FROM
eventtable
GROUP BY show_id
HAVING COUNT(show_id) > @cnum)
AND release_date = 2017
```

User

The user select shows and book tickets for the events of the shows he is interested in.

When the user enters the interface ,all the shows that are being held at the theater are displayed and the below query implements this

User has 2 main queries :

1. Inserting a new record in to customer table
2. Inserting new record in booking table

1. Insertion into customer

There are multiple queries involved here which finally lead up to the insert query. The other queries are mainly to get the data from the user to insert.

List of shows

select distinct show_name from shows;

When the user chooses a particular show the dates for the events of that selected show are displayed

List of dates for that show

```
SET @show_name = '%Lynnside%';
select date_event
from eventtable w
where show_id in (select show_id
                  from shows
                  where show_name like @show_name)
```

When user chooses the date

(Once you get the date selected from the user, set it for the variable @chosenDate)
The show_name,event time ,the date and the price of the ticket for that selected event are displayed.

```
SET @chosenDate='2017-12-05';
select s.show_name,e.time_event,e.date_event,e.ticket_price from
eventtable e
join shows s
on e.show_id = s.show_id
where s.show_id in (select show_id from shows where show_name like
@show_name) and e.date_event = @chosenDate;
```

Now when user submits - " confirm booking button"

When user hits the confirm booking button.The event_id for which the user is booking a ticket gets selected.

```
select e_id from eventtable e
join shows s
on e.show_id = s.show_id
where s.show_id in (select show_id from shows where show_name like
@show_name) and e.date_event = @chosenDate;
```

When a new customer is added :

```
insert into
customer(customer_name,phone_no,email_id,payment_details)
values(@customer_name,@phone_no,@email_id,@payment_details);
Var buffer = select max(customer_id) from customer order by
customer_id desc
```

Here customer_id is auto incremented and when a new customer is added the value of the last customer_id is fetched and the new customer_id is the last customer_id+1.

2. Insertion into booking.

The queries here help in populating the insert query for booking.

The total cost has to be calculated before inserting the record into booking, this is done by:

```
SELECT distinct ticket_price from eventtable where e_id = eid

And this ticket price is then multiplied by the number of tickets the user
wants to purchase.
```

```
set @e_id = event_id;
set @customer_id = "1";
set @num_tickets = "10";
set @price = "500.00";
set @booking_date = "2017-08-08";
set @booking_time = "03:05:00";
insert into
booking(e_id,customer_id,num_tickets,price,booking_date,booking_time
,booking_lable)
values(@e_id,@customer_id,@num_tickets,@price,@booking_date,@booking
_time,’YES’)
```

Indexing

For queries which does not involve primary key takes a longer time to execute. For example, to find the show_ids in increasing order which has a gap more than 365 days between reservation date and actual event date takes longer time to execute. In such cases it's efficient to use Indexing. Creating an index over reservation date (r_date) and event date (date_event) using BTree index will make the query run faster and improve performance.

Changes that were implemented

1. We have auto-incremented the values of the primary keys of the tables booking,customer and the eventtable.
   So every time a new customer,booking or event is created in the corresponding table the value of the ids are generated automatically and does not require manual input.

2. The queries has been modified.Some of the queries that were framed in phase 2 cannot get any user inputs,so the queries in phase 3 has been framed in a way for a given user input,the tuples that are fetched by the query is based on the input given by the user.,in our case it is the admin who gives the input to the database system.

3. A correlated query that was missing in phase 2 has also been added.

Video is available at
https://www.youtube.com/watch?v=ASg02X8B7UQ&feature=youtu.be