

## **FINAL PROJECT PHASE II – Databases**

**Sixiang Chen      Section 415**

**Jing   Liu      Section 315**

### **Description**

This project focuses on the database of the Broadway shows related information. Users can search for the Broadway shows and artists information based on the criteria that they are interested in. They also can search for available tickets. In our final project, we are going to design and implement this database based on the data obtained from the Internet.

## 1. Improvement compared to Phase I proposal

Firstly, we add the security problem. We design the function for existing users to log in and detecting the invalid user name. Also we can discriminate correct password from wrong password. Further more, we add the Sign up function for new users. Users can register a new user account.

Also we adapt the SQL procedure more than Phase I. We mainly realize the search process by using procedure. This change greatly improves the interaction between user and the database.

## 2. Data Source

The major source of the Broadway show and Artists information comes from the official website of Broadway (<http://www.broadway.com/>). We also look up to a Broadway show database website for some complementary information, which is the Internet Broadway Database (<http://www.ibdb.com/>). Since there are no open source relational tables, we have to create our own data and input the data we found from the above website manually using excel and transform to CSV files, and then we import the CSV formatted dataset into the tables we have created.

## 3. The platform we are using is Mysql and WAMPSERVER2.4.

## 4. User's guide

### **Set up the running environment**

- a) Un-compress the archive of all the code and data file
- b) Create a new database, ready to import all the dataset from sql file
- c) Open SQL document, run the three .sql files in order like:

Create table\_update.sql -> Import data\_update.sql -> Procedure\_update

So that the environment of database are well prepared for the project

- d) Open index.html located in Project document by web browser
- e) Reach the main page of our project.

### **Explore the main page**

a) Header:

A colorful background picture aiming to catch eyes, including an HOME image button on top to get back to this main page from anywhere.

A "broadway show" logo to clearly claim the purpose of our website.



b) Body:

Header: Includes two tags, the one which is implemented here is 'home'

Left-side: Displays a attracting poster for recently popular as well as classical broadway shows; The below part shows two latest news related to broadway shows

Right-side: Main searching bars for this website, including a textbox allowing user to search for the shows of their favorite artists, several selection parts enabling user to search for shows based on several searching conditions, and both the login and signup button to verify user's status.



### Search with the project

#### a) Search by the name of the artists

- i. User enters the artist name without any quote (Input as the hint is recommended)
- ii. Click the right-corner's Go button
- iii. All the shows played by the input artist are showing on the left side

#### b) Search by the single condition

- i. User choose any of one selection condition (Constraint to 'Select Type', 'Select Score', 'Select Price')
- ii. Click the Search button below
- iii. All the shows satisfied to the chosen single condition are

c) Search by the multi-condition

- i. User are allowed to choose 'Type', 'Score', 'Price', 'Theatre', 'Seat' and 'Date' all at the same time to search for a more specific result. Choosing partially may lead to empty result because any un-chosen condition will be treated as 'None' so no record in database will match that.
- ii. Click the Search button below
- iii. Input following the Hint is highly recommended

**Login / Signup into the website**

a) Login process

- i. Click the login button, a new page will pop up and allows users to enter UID and password
- ii. If the UID and password match the record in the database, which mean a valid user, the browser will show 'Login Success' in the left corner; If the UID and password don't match any record in the database, the window will show 'User doesn't exist. Please go back to home and Sign up again'.
- iii. A new sub-window pops out, containing a home button helping users to back to main page

b) Sign up process

- i. Click the signup button, a new page will pop up and allows users to enter new information including UID, username, password, age and gender
- ii. Upon a successful registration, the browser will show 'Sign Up successfully! Please go to home page!.' In the right corner
- iii. A new sub-window pops out, containing a home button helping users to back to main page



## 5. Our Major/Minor areas of specialization

Our major area or specialization is building a forms-based interface with sophisticated report generate. We design a very colorful and nice-looking webpage, which is pretty user-friendly. We add details like the pull-down menu for easier use and add the calendar for date selection, which gives users a more clear and direct way of making choices.

We also create shortcut keys of Home for user to go back the homepage easily and quickly.

Our Minor area of specialization is the security issue. We can testify if the input of a UserID and password if valid. If the UserID and password is not verified, our website will deny the access of the any of the personal information of our existing member. We also design the Sign Up function for new users to register.

## 6. Analysis of the limitation of our system

The main limitation we have is that the format of date we get from the webpage is not consistent with the format of date in Mysql. Due to limited time and knowledge, we failed to transform the format of date from website to the format in Mysql, so that we cannot fulfill the data exchange between the website and our database. As a result, we actually cannot search for information through the date criterion. That is the main limitation of our system.

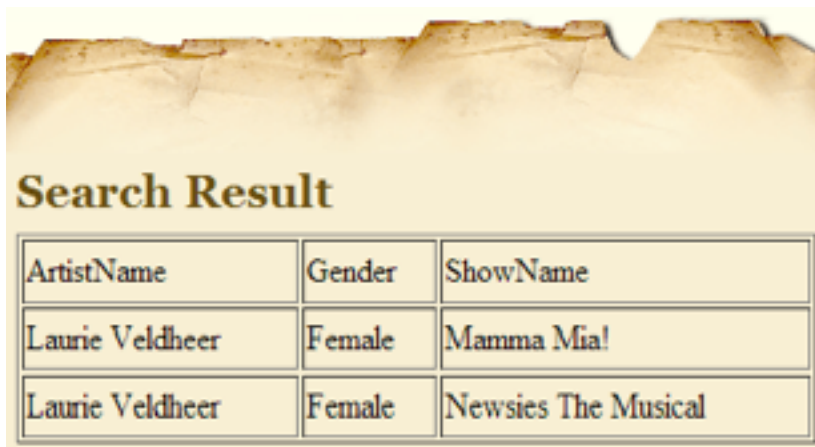
Also, we have limited combination of selection criterion. We cannot realize all the arbitrary queries raised by the users. For the pull-down menu, we cannot give answers to all the combination of the options.

Lastly, due to time and energy, also considering the way we import data to our dataset, which is manually input, our database is relatively small and not comprehensive.

For the above three limitations mentioned, if we invest more time into this, we can fix these limitations and improve our system.

## 7. Output of the project

The output generated from our website is aiming to satisfy people's general interests of Broadway shows. It is very common to see some people would like to follow their favorite artists and are intending to see as many shows played by those artists as they can. Thus we implements a separate textbox allowing users to get historical / latest records of the artists. On top of that, people are used to choose the shows they want to see according to the date, ticket price, type and the score made by professional reviewers. Thus our website offers several critical selection menus to help people sift information and narrow down their interests. Below are several typical outcomes generated from our database, which mainly focus on common user habits.

The image shows a screenshot of a web application interface. At the top, there is a header with a torn paper effect and the text "Search Result" in a bold, serif font. Below the header is a table with three columns: "ArtistName", "Gender", and "ShowName". The table contains two rows of data. The first row shows "Laurie Veldheer" as the artist, "Female" as the gender, and "Mamma Mia!" as the show. The second row shows "Laurie Veldheer" as the artist, "Female" as the gender, and "Newsies The Musical" as the show. The table has a simple border and is set against a light yellow background.

ArtistName	Gender	ShowName
Laurie Veldheer	Female	Mamma Mia!
Laurie Veldheer	Female	Newsies The Musical

Output for all the shows played by a specific artist



### Search Result

ShowID	ShowName
10001	The Lion King
10002	The Lion King
20001	The Phantom of the Opera
20002	The Phantom of the Opera
30001	Jersey Boys
30002	Jersey Boys
50001	Once
50002	Once
60001	First Date

Output for part of the shows for 'Musical' Type



### Search Result

ShowID	ShowName	ShowDate	StartingTime
20002	The Phantom of the Opera	2013-12-27	20:00:00
70002	Mamma Mia!	2013-12-27	20:00:00
120002	After Midnight	2013-12-27	20:00:00
170002	Kinky Boots	2013-12-27	20:00:00

Output for all the shows which is 'Musical' Type, has a score higher than 85, ticket price is lower than 200, played in Theater 'Minskoff Th', seated in 'Rear Mezzanine' and is going to play on 2013-12-27 20:00:00.

## 8. Relational Table

The relational tables are



USER	<u>UserID</u>	User Name	Last log in date	Gender	Age	Password
	Jliu88	Jing Liu	2013-11-25	Female	24	123456
	Schen97	Sixiang Chen	2013-09-09	Female	20	234567

BOOKED	<u>UserID</u>	<u>TicketID</u>	Book Time
	Jliu88	0010001B02	2013-11-12
	schen97	0260001B04	2013-12-20

TICKET	<u>TicketID</u>	Price	Seat Location
	0010001A01	255.25	Orchestra
	0050001B09	135	Front Mezzanine

HAS	<u>TicketID</u>	ShowID
	0070001C07	70001
	0060001A07	60001

LOCATION	<u>LID</u>	Theater	Address	ZipCode
	NYNY0001	Minskoff Theater	200 West 45	NY10036
	NYNY0016	Neil Simon	250 West 52	NY10036

PUT ON	<u>LID</u>	<u>ShowID</u>
	NYNY0001	10001
	NYNY0013	130002

TIME	<u>TimeID</u>	Date	Starting Time
------	---------------	------	---------------

	131227A	2013-12-27	20:00:00
	131222A	2013-12-12	15:00:00

SCHEDULED	<u>TimeID</u>	<u>ShowID</u>
	131221A	10001
	131225A	50001

SHOW TEMPLATE	<u>Show Number</u>	Show Name	Category	Duration	Rating Score
	1	The Lion King	Musical, kids-friendly	2 hrs, 30 minutes	95
	2	The Phantom of the Opera	Musical, Drama, Classics	2 hrs, 45 minutes	95

ARTIST	<u>AID</u>	Artist Name	Gender
	13501	Mary Michael Patterson	Female
	13502	Hugh Panaro	Male

CAST_IN	<u>AID</u>	<u>ShowID</u>
	13501	10001
	13502	10001

SHOW INSTANCE	Show Number	<u>ShowID</u>	Total Tickets	Available Tickets
	1	10001	30	29

## 9. SQL code (partially)

## Final Project

## Create Database

DROP TABLE IF EXISTS USER;

CREATE TABLE USER (UserID varchar(30), UserName varchar(30), Last\_login date, Gender varchar(6), Age int, Password char(6) not null, CONSTRAINT u\_pk PRIMARY KEY (UserID));

DROP TABLE IF EXISTS TICKET;

CREATE TABLE TICKET (TicketID char(17) primary key , Price float(4), SeatLocation varchar(20));

DROP TABLE IF EXISTS BOOKED;

CREATE TABLE BOOKED (UserID varchar(30) , TicketID char(10) , BookTime date, CONSTRAINT bk\_fk FOREIGN KEY (TicketID) REFERENCES TICKET(TicketID));

DROP TABLE IF EXISTS SHOW\_INSTANCE;

CREATE TABLE SHOW\_INSTANCE (ShowID int ,ShowNumber int, TotalTickets int, AvailableTickets int, CONSTRAINT si\_pk PRIMARY KEY (ShowID));

DROP TABLE IF EXISTS HAS;

CREATE TABLE HAS (TicketID char(10), ShowID int, CONSTRAINT h\_fk FOREIGN KEY (TicketID) REFERENCES TICKET(TicketID));

DROP TABLE IF EXISTS LOCATION;

CREATE TABLE LOCATION (LID char(8) primary key, Theater varchar(30), Address varchar(30), ZipCode varchar(10));

DROP TABLE IF EXISTS PUTON;

```
CREATE TABLE PUTON (LID char(8) , ShowID int, CONSTRAINT po_fk  
FOREIGN KEY (LID) REFERENCES LOCATION(LID));
```

```
DROP TABLE IF EXISTS SHOWTIME;  
CREATE TABLE SHOWTIME (TimeID char(7), ShowDate date, StartingTime time,  
CONSTRAINT st_pk PRIMARY KEY (TimeID));
```

```
DROP TABLE IF EXISTS SCHEDULED;  
CREATE TABLE SCHEDULED (TimeID char(7), ShowID int, CONSTRAINT  
sc_fk FOREIGN KEY (ShowID) REFERENCES SHOW_INSTANCE(ShowID));
```

```
DROP TABLE IF EXISTS SHOW_TEMPLATE;  
CREATE TABLE SHOW_TEMPLATE (ShowNumber int, ShowName varchar(35),  
Category varchar(20), Duration varchar(10), RatingScore int, CONSTRAINT sht_pk  
PRIMARY KEY (ShowNumber));
```

```
DROP TABLE IF EXISTS ARTIST;  
CREATE TABLE ARTIST (AID int, ArtistName varchar(30), Gender varchar(6),  
CONSTRAINT a_pk PRIMARY KEY (AID));
```

```
DROP TABLE IF EXISTS CAST_IN;  
CREATE TABLE CAST_IN (AID INT, ShowID INT);
```

```
##Import CSV file into Mysql
```

```
## Load SHOW_TEMPLATE
```

```
INSERT INTO `show_template` VALUES
```

```
  ('1', 'The Lion King', 'Musical, kids-friendly', '2hrs30mins', '95'),  
  ('2', 'The Phantom of the Opera', 'Musical, Drama, Classics', '2hrs30mins', '95'),  
  ('3', 'Jersey Boys', 'Musical, Original', '2hrs30mins', '90'),  
  ('4', 'Wicked', 'Kids-friendly', '2hrs45mins', '92'),  
  ('5', 'Once', 'Musical, Romance', '2hrs15mins', '90'),
```

```

('6', 'First Date', 'Musical, Comedy', '1hr30mins', '88'),
('7', 'Mamma Mia!', 'Musical, Comedy, Classics', '2hrs30mins', '95'),
('8', 'Chicago', 'Musical, Dance, Classics', '2hrs30mins', '92'),
('9', 'Richard III', 'Play, History', '2hrs50mins', '86'),
('10', 'A Gentleman\'s Guide to Love and Murder ', 'Comedy', '2hrs20mins', '90'),
INSERT INTO `location` (`LID`, `Theater`, `Address`, `Zipcode`) VALUES
('LID', 'Theater', 'Address', 'ZipCode'),
('NYNY0001', 'Minskoff Th', '200 West 45', 'NY10036'),
('NYNY0002', 'Majestic Th', '247 West 44', 'NY10036'),
('NYNY0003', 'August Wils', '245 West 52', 'NY10019'),
('NYNY0004', 'Gershwin Th', '222 West 51', 'NY10019'),
('NYNY0005', 'Bernard B. ', '242 W 45th ', 'NY10036'),
('NYNY0006', 'Longacre Th', '220 West 48', 'NY10036'),
('NYNY0007', 'Broadhurst ', '235 West 44', 'NY10036'),
('NYNY0008', 'Ambassador ', '219 West 49', 'NY10036'),
('NYNY0009', ' Belasco Th', '111 West 44', 'NY10036'),
('NYNY0010', ' Walter Ker', '219 West 48', 'NY10036'),
INSERT INTO `USER` VALUES
('jliu88', 'Jing Liu', '2013-11-28', 'Female', '24', '123456'),
('schen97', 'Sixiang Chen', '2013-12-20', 'Female', '20', '567890');

```

```

INSERT INTO `artist` VALUES
('13501', 'Mary Michael Patterson', 'Female'),
('13502', 'Hugh Panaro', 'Male'),
('13503', 'Jason Raize', 'Male'),
('13504', 'John Vickery', 'Male'),
('13505', 'Dominic Scaglione Jr.', 'Male'),
('13506', 'Drew Gehling', 'Male'),
('13507', 'Lindsay Mendez', 'Female'),
('13508', 'Alli Mauzey', 'Female'),
('13509', 'Paul Alexander Nolan', 'Male'),

```

```
('13510', 'Joanna Christie', 'Female'),  
('13511', 'Zachary Levi', 'Male'),  
('13512', 'Krysta Rodriguez', 'Female'),  
('13513', 'Judy Mclane', 'Female'),
```

```
.....
```

```
.....
```

```
##SQL PROCEDURE
```

```
-- QUERY 1
```

```
DROP PROCEDURE IF EXISTS Artist;
```

```
delimiter //
```

```
CREATE PROCEDURE Artist (IN ArtistName varchar(30))
```

```
BEGIN
```

```
SELECT DISTINCT A.ArtistName, A.Gender, ST.ShowName
```

```
FROM  Artist AS A INNER JOIN CAST_IN AS CI ON A.AID=CI.AID
```

```
      INNER JOIN SHOW_INSTANCE AS SI ON CI.ShowID=SI.ShowID
```

```
      INNER JOIN SHOW_TEMPLATE AS ST ON
```

```
ST.ShowNumber=SI.ShowNumber
```

```
WHERE  A.ArtistName=ArtistName;
```

```
END//
```

```
delimiter ;
```

```
# call Artist('Rafe Spall');
```

```
-- QUERY 2
```

```
DROP PROCEDURE IF EXISTS SearchType;
```

```
delimiter //
```

```
CREATE PROCEDURE SearchType (IN Type varchar(30))
```

```
BEGIN
```

```
SELECT DISTINCT SI.ShowID, ST.ShowName
FROM  SHOW_TEMPLATE AS ST INNER JOIN SHOW_INSTANCE AS SI ON
ST.ShowNumber=SI.ShowNumber
WHERE  ST.Category LIKE CONCAT("%",Type,"%");
END//
delimiter ;
#CALL SearchType ('PLAY');
```

-- QUERY 3

```
DROP PROCEDURE IF EXISTS SearchScore;
delimiter //
```

```
CREATE PROCEDURE SearchScore (IN Score int)
BEGIN
SELECT DISTINCT SI.ShowID, ST.ShowName
FROM  SHOW_TEMPLATE AS ST INNER JOIN SHOW_INSTANCE AS SI ON
ST.ShowNumber=SI.ShowNumber
      INNER JOIN SCHEDULED AS SCH ON SCH.ShowID=SI.ShowID
      INNER JOIN SHOWTIME AS SHT ON SHT.TimeID=SCH.TimeID
      INNER JOIN HAS AS H ON SI.ShowID=H.ShowID
      INNER JOIN TICKET AS T ON H.TicketID=T.TicketID
WHERE  ST.RatingScore>Score;
END //
```

```
delimiter ;
#CALL SearchScore (80);
```

-- QUERY 4

```
DROP PROCEDURE IF EXISTS SearchPrice;
delimiter //
```

```

CREATE PROCEDURE SearchPrice (IN Price float(4))
BEGIN
SELECT DISTINCT SI.ShowID, ST.ShowName
FROM  SHOW_TEMPLATE AS ST INNER JOIN SHOW_INSTANCE AS SI ON
ST.ShowNumber=SI.ShowNumber
      INNER JOIN SCHEDULED AS SCH ON SCH.ShowID=SI.ShowID
      INNER JOIN SHOWTIME AS SHT ON SHT.TimeID=SCH.TimeID
      INNER JOIN HAS AS H ON SI.ShowID=H.ShowID
      INNER JOIN TICKET AS T ON H.TicketID=T.TicketID
WHERE T.Price<=Price;
END //

```

delimiter ;

#CALL SearchScore (80);

-- QUERY 5

DROP PROCEDURE IF EXISTS FindShow;

delimiter //

```

CREATE PROCEDURE FindShow (IN Type varchar(30), IN Score int, IN Price
float(4), IN Seat varchar(20), IN Sdate date)
BEGIN

```

```

SELECT DISTINCT SI.ShowID, ST.ShowName, SHT.ShowDate, SHT.StartingTime
FROM  SHOW_TEMPLATE AS ST INNER JOIN SHOW_INSTANCE AS SI ON
ST.ShowNumber=SI.ShowNumber
      INNER JOIN SCHEDULED AS SCH ON SCH.ShowID=SI.ShowID
      INNER JOIN SHOWTIME AS SHT ON SHT.TimeID=SCH.TimeID
      INNER JOIN HAS AS H ON SI.ShowID=H.ShowID
      INNER JOIN TICKET AS T ON H.TicketID=T.TicketID

```



```
WHERE ST.Category LIKE CONCAT("%",Type,"%") AND ST.RatingScore>Score
AND T.Price<=Price AND T.SeatLocation=Seat AND SHT.ShowDate=Sdate;
END //
```

```
delimiter ;
```

```
#call FindShow ('musical', 80, 250.00, 'Rear Mezzanine','2013-12-27');
```

```
-- QUERY 6
```

```
DROP PROCEDURE IF EXISTS LOGIN;
```

```
delimiter //
```

```
CREATE PROCEDURE LOGIN (IN UID varchar(20),IN password varchar(10))
```

```
BEGIN
```

```
SELECT * FROM USER WHERE USER.UserID=UID AND
```

```
USER.Password=password;
```

```
end; //
```

```
delimiter ;
```

```
#call LOGIN ('jliu88', '123456');
```

```
-- QUERY 7
```

```
DROP PROCEDURE IF EXISTS REGISTER;
```

```
delimiter //
```

```
CREATE PROCEDURE REGISTER (IN UID varchar(20),IN UName varchar(30),IN
```

```
Gender varchar(6),IN Age int, IN password varchar(10))
```

```
BEGIN
```

```
IF EXISTS(SELECT * FROM USER WHERE USER.UserID=UID)
```

```
THEN SELECT "This UserID Already Exists";
```

```

ELSE
    INSERT INTO `USER` VALUES
    (UID,UName,CURDATE(), Gender, Age, password);
    SELECT "Successfully Registered";
END IF;
END; //

delimiter ;

#call REGISTER ('chunys91','Chun Yang', 'Female','22', '1q2w3e');

```

-- QUERY 8

```

DROP PROCEDURE IF EXISTS SearchTicket;
delimiter //

CREATE PROCEDURE SearchTicket (IN Sdate date)
BEGIN

    SELECT SI.ShowID, MIN(T.Price) AS MinimumPrice, SI.AvailableTickets,
    ST.ShowName
    FROM  SHOW_TEMPLATE AS ST INNER JOIN SHOW_INSTANCE AS SI ON
    ST.ShowNumber=SI.ShowNumber
        INNER JOIN HAS AS H ON SI.ShowID=H.ShowID
        INNER JOIN TICKET AS T ON H.TicketID=T.TicketID
        INNER JOIN SCHEDULED AS SCH ON SCH.ShowID=SI.ShowID
        INNER JOIN SHOWTIME AS SHT ON SCH.TimeID=SHT.TimeID
    WHERE  SHT.ShowDate=SDATE
    GROUP BY SI.ShowID
    limit 1;

```

END //

delimiter ;

#call SearchTicket ('2013-12-26');

-- QUERY 9

DROP PROCEDURE IF EXISTS Award;

delimiter //

CREATE PROCEDURE Award ()

BEGIN

SELECT ST.ShowName, A.ArtistName

FROM SHOW\_TEMPLATE AS ST INNER JOIN SHOW\_INSTANCE AS SI ON

ST.ShowNumber=SI.ShowNumber

INNER JOIN CAST\_IN AS CI ON CI.ShowID=SI.ShowID

INNER JOIN Artist AS A ON A.AID=CI.AID

WHERE ST.Category LIKE CONCAT("%","Tony Winners","%");

END//

delimiter ;