

CS2102 Report - A Movie Database Project

Vu Truong Vinh - U0605436
Tran Quoc Tuan - U0605132
Nguyen Anh Cuong - U0605126
Thai Ngoc Thuy Huong - U0605131
Nguyen Phuc Khanh Luan - U0605423

National University of Singapore, School of Computing

November 2, 2007

Contents

1	Server Information	2
1.1	Web Server	2
1.2	Database Management System	2
2	Database Model	3
2.1	ER Diagram	3
2.2	Relational Schema	3
2.3	Description	5
3	Implementation	7
3.1	Database Implementation	7
3.2	Webpage Implementation	7
	List of Figures	18

Chapter 1

Server Information

1.1 Web Server

Server Information :

- Server Name : http://cs2102-11-z
- Server Page Language : HTML, Javascript, PhP

1.2 Database Management System

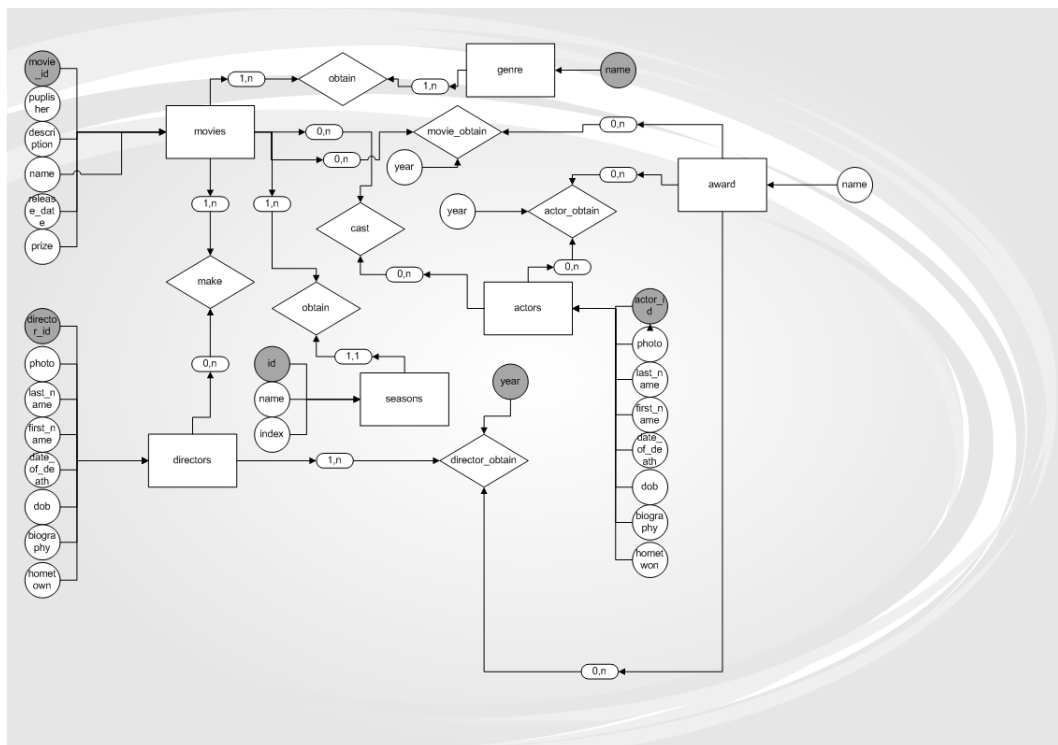
We use *sybase* as our database management system

Chapter 2

Database Model

2.1 ER Diagram

The ER Diagram is also included inside the report package



2.2 Relational Schema

Below is SQL code for our database relational schema

```
create table movies(  
    id numeric primary key not null default autoincrement,
```

```

        name varchar(64) not null,
        release-date Date,
        price numeric,
        publisher varchar(64),
        description varchar(1024)
    )

create table genre(
    name varchar(32) primary key
)

create table obtain-genre(
    genre varchar(32) references genre(name),
    id numeric references movies(id),
    primary key(id,genre)
)

create table awards(
    id numeric primary key not null default autoincrement,
    name varchar(32) not null,
    year numeric
)

create table movie-obtain-award(
    movie-id numeric references movies(id),
    award-id numeric references awards(id),
    category varchar(256),
    award-result varchar(16),
    primary key (movie-id, award-id)
)

create table actors(
    id numeric primary key not null default autoincrement,
    first-name varchar(32) not null,
    last-name varchar(32) not null,
    gender varchar(8),
    dob Date,
    dod Date,
    hometown varchar(32),
    biography varchar(1024),
    photo varchar(32)
)

create table movies-cast(
    movie-id numeric references movies(id),
    actor-id numeric references actors(id),
    role varchar(32),
    primary key (movie-id, actor-id)
)

create table actor-obtain-award(
    actor-id numeric references actors(id),

```

```

award-id numeric references awards(id),
category varchar(256),
result varchar(16),
primary key (actor-id, award-id)
)

create table seasons(
id numeric primary key not null default autoincrement,
season-index numeric not null
)

create table contain(
movie-id numeric references movies(id),
season-id numeric references seasons(id),
primary key (movie-id,seasonid)
)

create table directors(
id numeric primary key not null default autoincrement,
first-name varchar(32) not null,
last-name varchar(32) not null,
gender varchar(8),
dob Date,
dod Date,
hometown varchar(32),
biography varchar(1024),
photo varchar(32)
)

create table make(
movie-id numeric references movies(id),
director-id numeric references directors(id),
primary key (movie-id, director-id)
)

create table director-obtain-award(
director-id numeric references directors(id),
award-id numeric references awards(id),
category varchar(256),
award-result varchar(16),
primary key (director-id, award-id)
)

```

2.3 Description

There are three main tables : *directors*, *actors* and *movies*, three sub-tables : *award*, *seasons* and *genre* and some relation tables : *make*, *cast*, *movie-obtain-seasons*, *movie-obtain-genre*, *movie-obtain-award*, *actor-obtain-award*, *director-obtain-award*

This database stores information about directors, actors and movies. Directors and Movies, Actors and Movies are linked together thorough some relation

table so that the user can find out which directors make which movies, which actors cast which movies and many other related informations. For example : the film Jurassic Park is made by the director Stephen Speilberg then the film Jurassic Park is stored in movies table with an id, for instance id 10, and the director Stephen Speilberg is stored in directors table with an id, for instance id 2. And two ids will be stored in a row of the *make* table to indicate that the director with id 2 make the film with id 10. A similar convention is used for the relation between movies and actors.

The database also records many other information of movies, actors and directors such as : award for movies, actors and directors, season and genre for movies.

We use only one table award to record the award for all movies, actors and directors instead of use three distince tables. This convention reduces the number of tables and also simplify the database model. The year of obtaining award are also put in the relation tables: *actor-obtain-award*, *director-obtain-award* and *movie-obtain-award* because it reduces the size of the award table and make it easy to maintain and update.

The genre are stored in a different table instead of putting into the movies table because a movie can have many genre, therefore putting the genre into movies table will increase the size of this table.

Since a movie also can have many seasons, the table season are constructed for this purpose.

Chapter 3

Implementation

3.1 Database Implementation

We had two methods to import data into the the database:

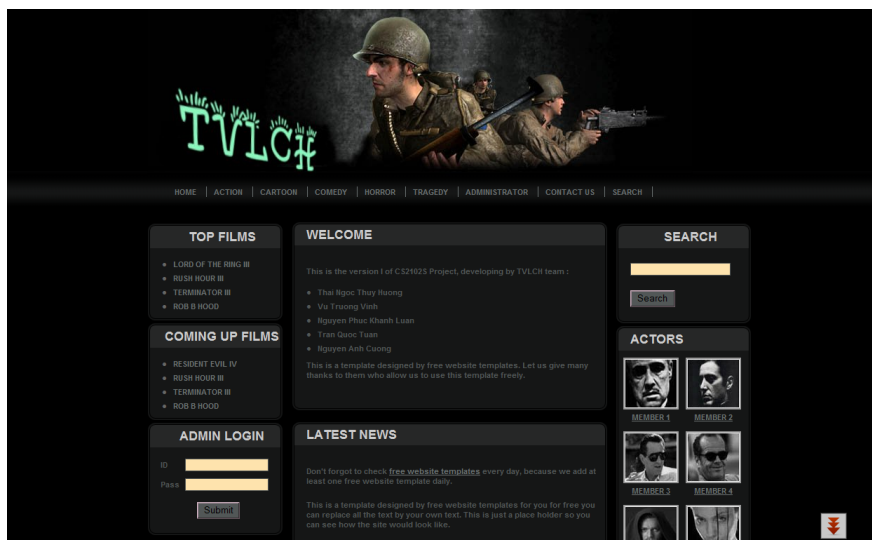
- Store the information in the text file with a simple format and use Java to convert into SQL codes
- Use excel file to import into database, with the supporting of SQL Anywhere 10 Interaction

Finally we use the second method to import data, because it save lots of time and safer

3.2 Webpage Implementation

In this section, we will discuss about the implementation of user interface and admin interface.

A. User Interface



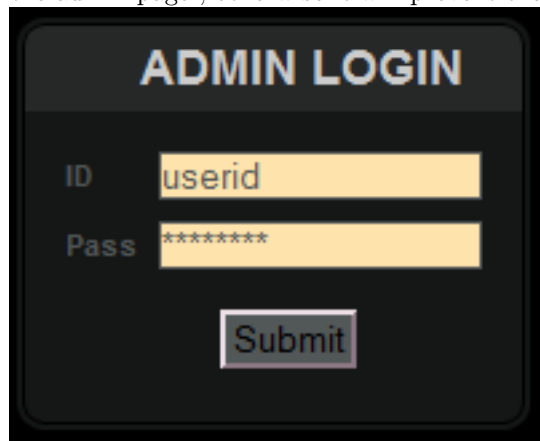
- We have three main modules in user interface which is : admin login, advance search and normal search (google style)
- Admin Login Module
 - We stored the useid and password in php code using session so that the user can not see this information

```

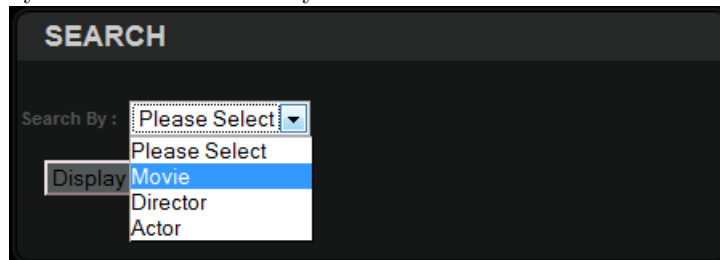
<?php
include ("generic.php");
//session_start(); // start up your PHP session!
$log = is_login();
if($_POST || $log){
  if($log || ($_POST['user']=='u0605131') && ($_POST['password']=='sybase123')){
    $_SESSION['user']='u0605131';
    $_SESSION['password']='sybase123';
    include("admin.html")
  }else echo "<center><B><h3>YOUR PASSWORD DOES NOT MATCH WITH YOUR USERNAME</h3></B></center>.".
    "<center><b><a href='index.php'>CLICK HERE</a> TO TRY AGAIN.</b></center>";
  }else echo "<center><B><h3>YOU HAVEN'T LOGIN YET! YOU NEED TO LOG IN AGAIN.</h3></B></center>.".
    "<center><b><a href='index.php'>CLICK HERE</a> TO TRY AGAIN.</b></center>";?>

```

- When the user enters correct id and password, then he/she can go to the admin page , otherwise it will prevent the user go to that page



- Advance Search Module
 - We have three option for searching which is : search by movies, search by directors and search by actors



- The reason for us to separate this search into three option because it is more useful and easy to use for the user
- For every option we used different filter boxs. And also for preventing the anoying for the users when choosing a option, we use javascript

so that the page is not refreshed

```

<script>
function search_display(type){
var para = document.getElementById("display_search");
if (type == 'movie'){
para.innerHTML = '<br>';
para.innerHTML += '<form name="searchdirector" method="post" action="#"><table border="0" cellspac
}
else if (type == 'director'){
para.innerHTML = '<br>';
para.innerHTML += '<form name="searchdirector" method="post" action="#"><table border="0" cellspac
}
else if (type == 'actor'){
para.innerHTML = '<br>';
para.innerHTML += '<form name="searchdirector" method="post" action="#"><table border="0" cellspac
}
}
}
</script>
<table>
<tr>
<td> Search By :</td>
<td>
<select onChange="search_display(this.value)">
<option selected> Please Select &nbsp;   </option>
<option name="movie" value="movie"> Movie </option>
<option name="director" value="director"> Director </option>
<option name="actor" value="actor"> Actor </option>
</select>
</td>
</tr>
</table>
<div id="display_search"></div>

```

- The movies search module can search by name, release date, publisher, price, rating, director, actor and award . If the information is not given then we will ignore all, this implies that if the user did not give any information then the system will display all the data in movie table

- The information will display according to the filter given when the user click the display button. However, all the information the user typed in are still kept for reference

- Normal Search Module

- This will be implemented using Google style, in the sense that the user can type any keyword at the search box and the system will display all informations related

- For the text given, the system will split in separate strings and search for this string is all attributes of all tables : directors, movies, actors and take the union the there result to display

- There are some another minor features in the user inteface which is

- There are two side bars on the left hand side of the page which is : top films and coming up films which is aslo implemented using the database

```

<?php include("generic.php");
$sql="select name from movies order by release_date desc";
$result = rowarray($sql);
$i = 0;
foreach ($result as $index){
    if ($i < 5){
        echo "<li>$index[name]</li>";
    }
    $i++;
}
?>

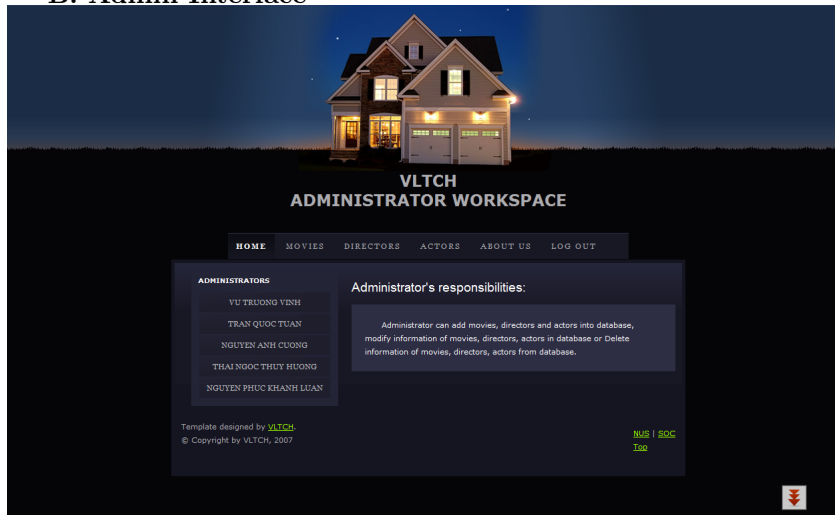
```

- When the user click on the action tab, the system will display top ten action films. Similarly for cartoon, comedy, tragedy and horror film

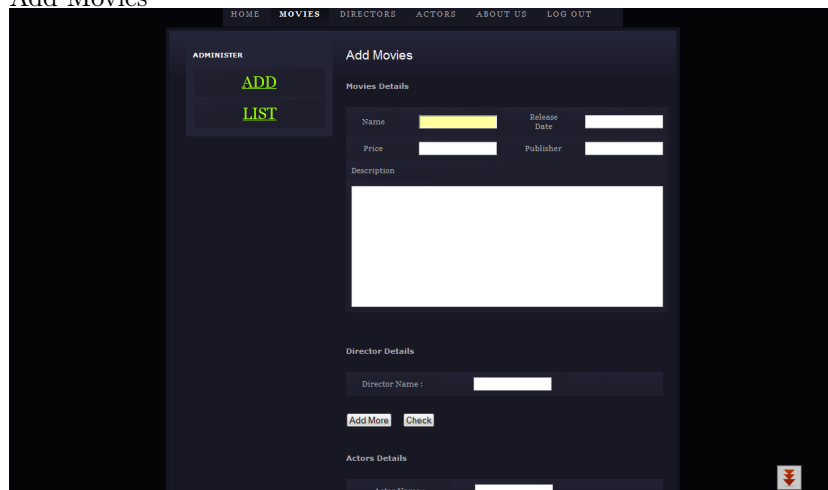
ACTION | CARTOON | COMEDY | HORROR | TRAGEDY

- When the user click on a film or an actor/director picture, the system will also display all the information of the movie/actor/director chosen

B. Admin Interface



- The admin page was implemented so that the user can not go directly to the page without login
- There are three main modules in the admin interface : add/list-edit movies, add/list-edit directors and add/list-edit actors . Following is the description for three sections : add movies, list/edit movies. The other two modules are implemented similarly
- Add Movies



- The system will record all the information it needs for the database, include of : movie name, movie release date, movie price, movie pub-

lisher and movie description

```
$name = $_POST['name'];
$releasedate = $_POST['releasedate'];
$description = $_POST['description'];
$price = $_POST['price'];
$publisher = $_POST['publisher'];
$sql = "insert into movies (name, release_date, price, publisher, description)
      values ('.$name.', '$releasedate.', '$price.', '$publisher.', '$description)";
if (isset($name)){
    execute($sql);
}
```

- The system also records the directors and actors participating in the movie. There are two buttons "add more" and check inside the director section and actor section. The add more button is used when the user want to record many directors and actors. When the user click add more, another box directors/actors name will appear. The check button is used for safe record purpose. Before the user click submit, he/she should use the check button if he/she is not sure about the director/actor name he/she types in. The director/actor must be in the database already before it can be used. When the user click on the check button, an alert will appear beside the director name : exist or not exist. In the case of exist, the system will list all the director with similar name so that the user can choose the correct information he/she wants. In the case the user use wrong director/actor name and click submit. the system will deny and not allow the information to be recorded

Director Details

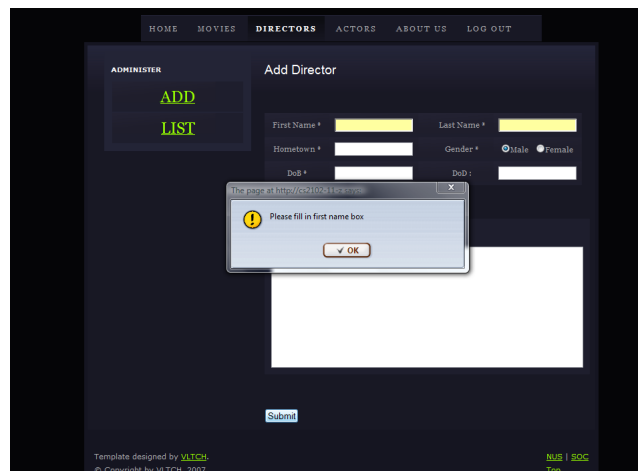
Director Name :

Add More Check

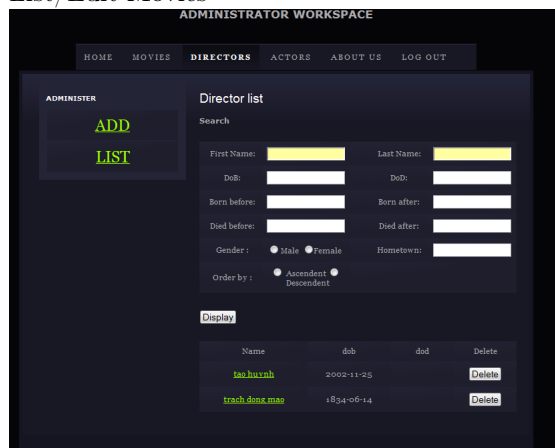
Actors Details

Actor Name :

Add More Check



- List/Edit Movies



- The user need to type in some informations to help the system display the movies that the user want. The option given is: name, publisher, price, release date, description and order by. All these filter boxes are optional, and in case no information is given, the system will display all records


```

|<?php
|if($result){
|echo
|'<table width="822" border="0" cellspacing="0">
|<td width="161">movie name</td>
|<td width="162">release date </td>
|<td width="91">price</td>
|<td width="82"> Delete </td>
|</tr>';
|foreach($result as $row){
|?><tr>
|<td><a href="movies.php?action=list&id=<?php echo $row[id] ?>"><?php echo $row[name]?></a></td>
|<td><?php echo $row[release_date] ?></td>
|<td><?php echo $row[price] ?></td>
|<td><input type="button" value="Delete" onclick="location.href='movies.php?delete_id=<?php echo $row[id]
|&release_date=<?php echo $release_date ?>&price=<?php echo $price ?>'"/></td>
|</tr>
|<?php }echo"</table>";}else if($display!=0)echo "<br/>No Result match with your query";
|?>

```

- After the system display movies list, the user will have the option to edit the information of movies or delete it. To edit, the user click on the name of movies and to delete the user click on the delete button beside the movie information

movie name	release date	price	Delete
Tinh Vo Mon3	2011-11-07	234.000000	Delete
Ta la ta	1986-12-29	67.000000	Delete
Tinh Vo mon1	2001-12-11	34.000000	Delete
sdfsff	2000-10-24	83.000000	Delete
kjefdksh	2000-10-24	41.000000	Delete
Tinh Vo Mon2	1999-06-25	13.000000	Delete
forest gump	1995-04-21	65.000000	Delete

```

if($_GET){
    if(isset($_GET['delete_id'])){
        $id = $_GET['delete_id'];
        execute("delete from movies where id = ".$id." ");
        $movie_name=$_GET['movie_name'];
        $publisher=$_GET['publisher'];
        $price =$_GET['price'];
        $release_date =$_GET['release_date'];
        $description=$_GET['description'];
        $order=$_GET['order'];
        $display=1;
    }
}

```

List of Figures

This page includes the code for the generic file, core function of the database:

```
<?php
function is_login(){
    session_start(); // start up your PHP session!
    //ob_start();
    if(isset($_SESSION)){
        if($_SESSION['user']=='u0605131' && $_SESSION['password']=='sybase123') {echo $_SESSION['user']=='u0605131';}
    }
    return false;
}
function getConn()
{
    $config = array("localhost","root","","cs2102");
    $conn = mysql_connect($config[0],$config[1],$config[2]);
    mysql_select_db($config[3], $conn);
    if (!$conn)
        throw new Exception("Unable to connect to database",10);
    return $conn;
}
function rowarray($sql){
    $conn=getConn();
    $res = mysql_query($sql);
    while ($row = mysql_fetch_array($res)){
        $result[] = $row;
    }
    return $result;
}
function execute($sql){
    $conn=getConn();
    mysql_query($sql);
}
};>
```