University of British Columbia, Department of Computer Science

CPSC 304

2016 Winter Term 2

Project Part _FINAL___

Group Name: Sineplex

Group Members:

Name	Student Number	Unix ID	Email Address
Zijing Lu	31463145	i2v9a	lizlu95@hotmail.com
Patrick Tseng	31859127	l6o8	l6o8@ugrad.cs.ubc.ca
Haoming Sun	26219097	m7r7	m7r7@ugrad.cs.ubc.ca
Xi Yang	46354122	r8z9a	melodyxi23@gmail.com

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above.
In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia.

List of queries: (we also show the query being used directly on the interface).

1. **(1 point) Selection and projection query:**

   Select ShowId, [Additional Headers] FROM arrange as a, theater as t, [ticket as ti] WHERE a.Location = t.Location and ti.ArrangeId = a.ArrangeId and [additional options]

2. **(1 point) Join query:**

   SELECT a.ArrangeId as ShowID, t.Name as Theater, a.Name, a.SeatsLeft as Seats from arrange as a inner join theater as t on a.Location = t.Location where SeatsLeft

3. **(1 point) Division query:**

   List users that bought a specific movie.

4. **(1 point) Aggregation query:**

   SELECT a.ArrangeId as ShowID, t.Name as Theater, a.Name, a.SeatsLeft as Seats from arrange as a inner join theater as t on a.Location = t.Location where SeatsLeft >/< (SELECT avg(SeatsLeft) from arrange);

5. **(2 points) Nested aggregation with group-by:**

   Select w.Location, " . $_POST['statsSeatsPerTheaterMinMax'] . "(w.m) as AvgSeatsLeft from (SELECT t.Location, avg(a.SeatsLeft) as m from theater as t, arrange as a where a.Location = t.Location GROUP BY t.Location) as w;

6. **(1 point) Delete operation:**

- Deleting user cascades down to their tickets
- Deleting arranges doesn't cascade to tickets, must delete tickets first
- Deleting movie doesn't cascade to tickets, must delete tickets first
- Deleting theaters cascades down to arranges

8. (1 point) Update operation:

- **Check seats before buying**
- **Check constraint added to creation of Arranges**

(2 points) Graphical user interface:

Done

7. **(3 points) Extra features:**
   a. **Giant database**
   b. **Ability to reset database**
   c. **Hosted on a webserver**

Project Accomplishment:

- Users able to search and buy tickets for movies with specific show times.
- Basic movie theater functionality
- Employees had more advanced searching + able to modify the database using an interface, e.g delete anything they wanted.

Things changed:

- Had to add a unique ID to arranges to make it easier to buy tickets for
- Added some CHECK constraints.
- Added theater IDs to make it easier to cross reference (but location being unique would have been fine)
- Didn't end up using movie.duration, movie.type and auditorium (too many extras).

Theater:
TheaterID -> Location, Name, OpenTime, CloseTime
Location -> TheaterID, Name, OpenTime, CloseTime
Descrip: Having TheaterID/Location determines attributes of a theater

Name -> Type, Duration, StartingDate, Price
Descrip: Having Name determines movie.

ID_> Name, EEmail, EPassword
Descrip: ID determines employee

CEmail -> Age, CPassword
Descrip: CEmail determines customer

ArrangeID -> ShowTime, Location, Name, SeatsLeft
Descrip: arrangeID determines the Arrange, has constraints on SeatsLeft >=0

ConfirmationNo -> SeatsNo, AuditoriumNO, CEmail, ArrangeId
Descrip: ConfirmationNo determines the unique ticket, and also determines the specific Arrange