CPSC 304 Project Cover Page

Milestone #: 4

Date: Nov 25th, 2024

Group Number: 29

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Ziqing Wang	22270649	g0h7c	g0h7c@ugrad.cs.ubc.ca
Owen Zheng	35933183	y6i3e	zcc2280411284@gmail.com
Jiawei Hu	57536633	i5m2m	i5m2m@ugrad.cs.ubc.ca

1. The domain of the application is ticket management & logistics for organizing concerts. This domain models ticketing services in the setting of successfully running concerts around the world (by providing locations). A real-life example is UBC’s on-campus concerts where people could order tickets online.
   The project focuses on the admin site, where we can manage and view user’s purchase histories and how they purchased tickets. There are also advanced filterings implemented to filter tickets with multiple tables joined.
2. The schemas remain the same and stick closely with ER diagrams. There is an assertion required for Concert to be able to satisfy the total participation constraint with the Attend_Locate table, but Oracle does not support assertion.
3. SQL Script sheet can be found in **./CPSC304_Node_Project-main/m4_sql.sql**
4. Below are all SQL queries with line number included: (All of the below queries are in the file appService.js)

2.1.1:

Insert (115):

```
`INSERT INTO TPH1 (seatnumber, cid, paymentmethod, paymentlocation, email, seatlocation)
VALUES (:seatnumber, :cid, :paymentmethod, :paymentlocation, :email, :seatlocation)`
```

2.1.2:

Update (134):

```
`UPDATE TPH1 SET paymentmethod = :paymentmethod, paymentlocation = :paymentlocation, email = :email, seatlocation = :seatlocation
where seatnumber = :seatnumber AND cid = :cid`
```

2.1.3:

Delete (148):

```
`DELETE FROM TPH1 WHERE seatnumber = :seatnumber AND cid = :cid`
```

2.1.4:

SELECTION (244):

```
SELECT * FROM TPH1 WHERE ${parsedString}
```

2.1.5:

PROJECTION (251):

```
`SELECT ${s} FROM Concert`
```

2.1.6:

Join (162):

```
`SELECT t.seatnumber, t.seatlocation
 FROM TPH1 t, Concert c
 WHERE t.cid = c.cid AND c.title = '${title}'
```

2.1.7:

GROUP BY (180):

```
`SELECT c.title, COUNT(*)
 FROM TPH1 t, Concert c
 WHERE t.cid = c.cid
 GROUP BY c.title
```

Meaning: This query is aimed to find the number of tickets sold for each concert.

2.1.8:

HAVING (199):

```
`SELECT a.email, COUNT(*)
 FROM Audience a, TPH1 tph
 WHERE a.email = tph.email
 GROUP BY a.email
 HAVING COUNT(*) >= 1
```

Meaning: This query is aimed to find the people who purchase at least one ticket.

2.1.9:

Nested Query (262):

```
`SELECT TPH1.email FROM TPH1, TPH2
```

```
WHERE TPH1.seatlocation = TPH2.seatlocation
GROUP BY email
HAVING AVG(price) > (SELECT AVG(price)
                     FROM TPH1, TPH2
                     WHERE TPH1.seatlocation = TPH2.seatlocation)`
```

Meaning: This query is aimed to find people whose average purchased price is greater than the price average of all concerts tickets.

2.1.10:

DIVISION (219):

```
`    SELECT a.email, a.audiencename
     FROM Audience a
     WHERE NOT EXISTS
         ((SELECT c.cid FROM Concert c)
         MINUS
         (SELECT t.cid FROM TPH1 t WHERE a.email = t.email))
```

Meaning: This query is aimed to find audiences who purchased tickets for every concert.