CPSC 304 Project Cover Page

Milestone #: ___2___

Date: __Oct. 25, 2021___

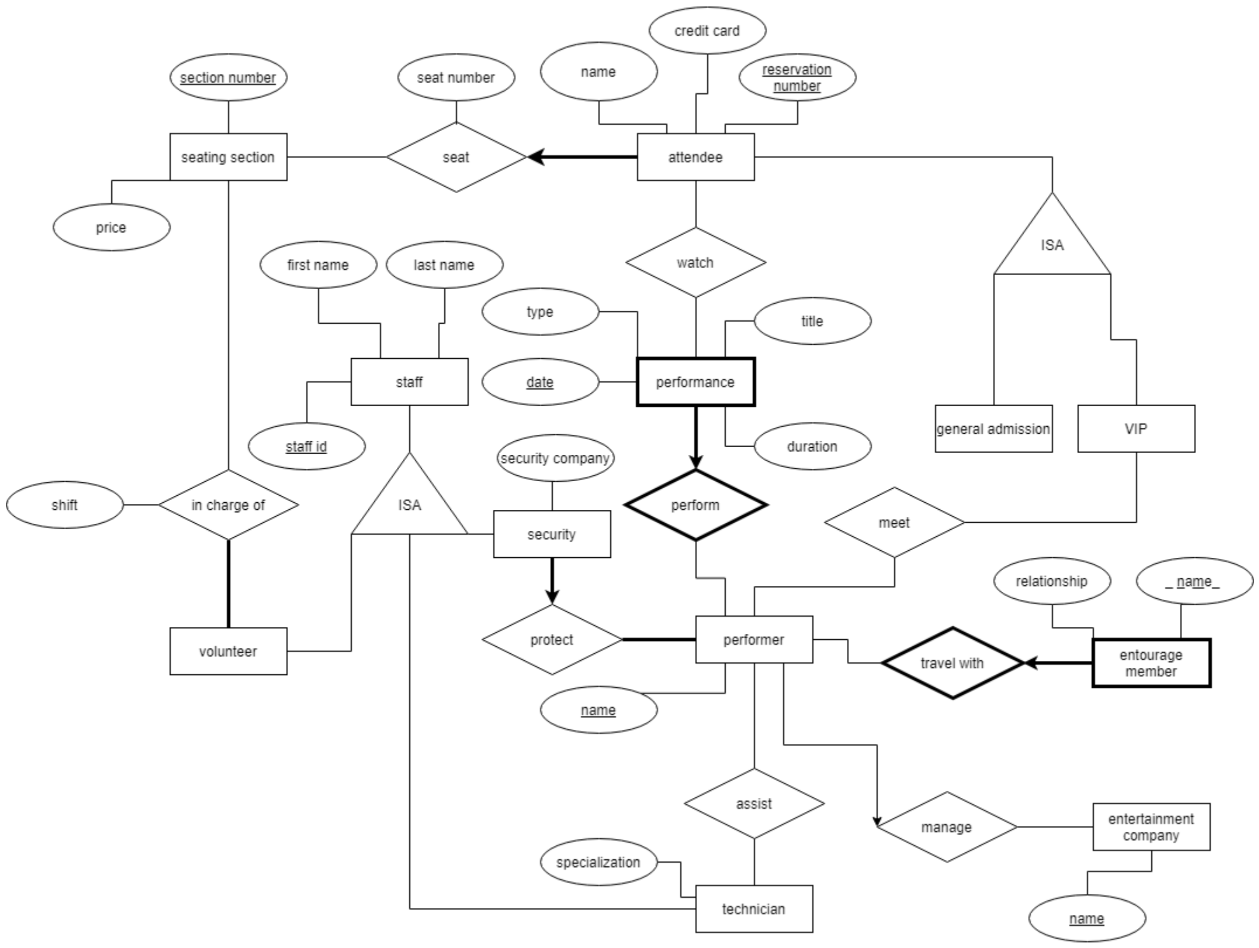
Group Number: ____14_____

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Doris Sun	41134776	d1w5a	doris.sun@queensu.ca
Raymond Zhang	86395571	s8b3b	raymondz6011@gmail.com
Michelle Kim	55441778	k3x2b	yeojin011016@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 the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

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.

ER Diagram:



Note: We have updated our diagram following Milestone 1 with the following changes:

1) The seat relationship between an attendee and a seating section was previously many-to-many, but we have changed this to be one-to-many and have changed the seat number to be an attribute of the seat relationship instead of an attribute and key for the seating section. It makes more sense for a single reservation number to be associated with a single seat in a single section, and a section can seat many attendees. We also added a credit card attribute that is unique to each attendee but is not a key, which makes the table in 2NF.

2) The protect relationship between security and a performer was previously many-to-many, but we have updated this to be one-to-many. This reflects the fact that a performer can be protected by more than one security guard, but a security guard is typically only assigned to one performer. We have kept the total participation constraint as previously designed.

3) We previously had a road crew as a weak entity to a performer, but the relationship that we had designed was not intuitively one-to-many. The weak entity was also part of a ternary relationship involving a performer and an entertainment company, when the management relationship did not need to involve the weak entity at all. As such, we have changed the weak entity to be an entourage member who travels with the performer while the performer is on tour. In reality, these entourage members could be the performer's family and friends, each having some sort of relationship to the performer, and each having a unique name, which we have chosen to be the weak entity's partial key.

4) We changed the key of performance from title to date, as the performer and date would be enough to identify a performance. Additionally, this adds a table that is only in 2NF.

Schema:

Note: For reference, primary keys are underlined, foreign keys bolded, candidate keys italicized, and other constraints are explicitly stated.

Entities:

SeatingSection(sectnum: integer, price: integer)
- sectnum is a primary key

GeneralAdmission(**resnum**: integer)
- resnum is a primary key and references Attendee_Seat

VIP(**resnum**: integer)
- resnum is a primary key and references Attendee_Seat

Staff(staffID: integer, firstname: string, lastname: string)

Volunteer(**staffID**: integer)
- staffID is a primary key and references Staff

Technician(**staffID**: integer, specialization: string)
- staffID is a primary key and references Staff

Security(**staffID**: integer, company: string)
- staffID is a primary key and references Staff

EntertainmentCompany(name: string)

Performer(name: string**,** **cname**: string)
- cname is a foreign key and references EntertainmentCompany

Weak Entities:

Travel_Entourage(ename: string, relationship: string, **name**: string)
- name is part of primary key and references Performer

Performance(date: string, type: string, title: string, duration: integer, **name**: string)
- name is part of primary key and references Performer

Relationships:

Attendee_Seat(resnum: integer, name: string, cardnum: integer, seatnum: integer, **sectnum**: integer)

- Participation constraint: seatnum and sectnum cannot be null
- sectnum is a primary key and references SeatingSection

In_Charge_Of(shift: string, **staffID**: integer, **sectnum**: integer)

- Participation constraint: sectnum cannot be null

Protect(**staffID**: integer, **protect-name**: string)

- staffID is a primary key and references Security
- protect-name is a primary key, references Performer, and cannot be null

Assist(**staffID**: integer, **assist-name**: string)

- staffID is a primary key and references Technician
- assist-name is a primary key and references Performer

Watch(**resnum**: integer, **name**: string, **date**: string)

- resnum is a primary key and references Attendee_Seat
- name is a primary key and references Performer
- date is a primary key and references performance

Meet(**resnum**: integer, **name**: string)

- resnum is a primary key and references Attendee_Seat
- name is a primary key and references Performer

Functional Dependencies (FDs):

SeatingSection

- sectnum → price, resnum

Attendee_Seat

- resnum → name, sectnum, seatnum, cardnum
- cardnum → name, sectnum, seatnum

GeneralAdmission

VIP

Staff

- staffID → firstname, lastname

Volunteer

Technician

- staffID → specialization

Security

- staffID → company

Performer

TravelEntourage

- ename, name→ relationship

Performance

- name, date → title, duration, type
- title → duration, type

In_Charge_Of

Protect

Assist

Watch
Meet

Normalization:
Attendee_Seat(resnum: integer, name: string, cardnum: integer, seatnum: integer, **sectnum**: integer)
Normalizes to:
Attendee_Seat_1(resnum: integer, cardnum: integer)
Attendee_Seat_2(name: string, cardnum: integer, seatnum: integer, **sectnum**: integer)

Performance(date: string, type: string, title: string, duration: integer, **name**: string)
Normalizes to:
Performance_1(date: string, title: string, **name**: string)
Performance_2(type: string, title: string, duration: integer)

All other tables are already in BCNF and 3NF.

SQL DDL:
```
CREATE TABLE SeatingSection(
        sectnum INTEGER,
        price INTEGER,
        PRIMARY KEY (sectnum))

CREATE TABLE Attendee_Seat_1(
        resnum INTEGER,
        cardnum INTEGER NOT NULL,
        PRIMARY KEY (resnum),
        FOREIGN KEY (cardnum ) REFERENCES
Attendee_Seat_2
        ON DELETE CASCADE
        ON UPDATE CASCADE)

CREATE TABLE Attendee_Seat_2(
        cardnum INTEGER,
        name CHAR(20),
        sectnum INTEGER NOT NULL,
        seatnum INTEGER NOT NULL,
        PRIMARY KEY (cardnum),
        FOREIGN KEY (sectnum) REFERENCES
SeatingSection
        ON DELETE CASCADE
        ON UPDATE CASCADE)
```

```
CREATE TABLE GeneralAdmission(
        resnum INTEGER
        PRIMARY KEY (resnum),
        FOREIGN KEY (resnum) REFERENCES
Attendee_Seat
        ON DELETE CASCADE
        ON UPDATE CASCADE)

CREATE TABLE VIP(
        resnum INTEGER
        PRIMARY KEY (resnum),
        FOREIGN KEY (resnum) REFERENCES
Attendee_Seat
        ON DELETE CASCADE
        ON UPDATE CASCADE)

CREATE TABLE Staff(
        staffID INTEGER,
        firstname CHAR(20),
        lastname CHAR(20),
        PRIMARY KEY (staffID))

CREATE TABLE Volunteer(
        staffID INTEGER,
        PRIMARY KEY (staffID),
        FOREIGN KEY (staffID) REFERENCES
Staff
        ON DELETE CASCADE
        ON UPDATE CASCADE)

CREATE TABLE Technician(
        staffID INTEGER,
        specialization CHAR(20),
        PRIMARY KEY (staffID),
        FOREIGN KEY (staffID) REFERENCES
Staff
        ON DELETE CASCADE
        ON UPDATE CASCADE)

CREATE TABLE Security(
        staffID INTEGER,
        company CHAR(20),
        PRIMARY KEY (staffID),
        FOREIGN KEY (staffID) REFERENCES
Staff
        ON DELETE CASCADE
        ON UPDATE CASCADE)

CREATE TABLE EntertainmentCompany(
        name CHAR(20)
        PRIMARY KEY (name))

CREATE TABLE Performer(
        name CHAR(20)
        cname CHAR(20)
        PRIMARY KEY (name),
        FOREIGN KEY (cname) REFERENCES
EntertainmentCompany
        ON DELETE CASCADE
        ON UPDATE CASCADE)

CREATE TABLE Travel_Entourage(
        ename CHAR(20),
        relationship CHAR(20),
        name CHAR(20)
        PRIMARY KEY (ename, name),
        FOREIGN KEY (name) REFERENCES
Performer
        ON DELETE CASCADE
        ON UPDATE CASCADE)

CREATE TABLE Performance_1(
        date CHAR(20),
        title CHAR(20),
        PRIMARY KEY (name, date),
        FOREIGN KEY (name) REFERENCES
Performer
        ON DELETE CASCADE
        ON UPDATE CASCADE,
        FOREIGN KEY (title) REFERENCES
Performance_2
        ON DELETE CASCADE
        ON UPDATE CASCADE)

CREATE TABLE Performance_2(
        title CHAR(20),
        duration INTEGER,
        name CHAR(20)
        PRIMARY KEY (title))

CREATE TABLE In_Charge_Of(
        staffID INTEGER,
        sectnum INTEGER,
        shift CHAR(20),
        PRIMARY KEY (staffID, sectnum),
        FOREIGN KEY (staffID) REFERENCES
Staff
        ON DELETE SET NULL
        ON UPDATE CASCADE,
        FOREIGN KEY(sectnum) REFERENCES
SeatingSection
        ON DELETE CASCADE
        ON UPDATE CASCADE)

CREATE TABLE Protect(
        staffID INTEGER,
        protect-name CHAR(20)
        PRIMARY KEY (staffID, protect-name),
        FOREIGN KEY (staffID) REFERENCES
Security
        ON DELETE CASCADE
        ON UPDATE CASCADE,
        FOREIGN KEY (protect-name) REFERENCES
Performer
        ON DELETE SET NULL
        ON UPDATE CASCADE)

CREATE TABLE Assist(
        staffID INTEGER,
        assist-name CHAR(20)
        PRIMARY KEY (staffID, assist-name),
        FOREIGN KEY (staffID) REFERENCES
Technician
        ON DELETE CASCADE
        ON UPDATE CASCADE,
        FOREIGN KEY (assist-name) REFERENCES
Performer
        ON DELETE CASCADE
        ON UPDATE CASCADE)

CREATE TABLE Watch(
        resnum INTEGER,
        pname CHAR(20),
        date CHAR(20),
        PRIMARY KEY (resnum, pname, date),
        FOREIGN KEY (resnum) REFERENCES
Attendee_Seat
        ON DELETE CASCADE
        ON UPDATE CASCADE,
        FOREIGN KEY (pname) REFERENCES
Performer
        ON DELETE CASCADE
        ON UPDATE CASCADE,
        FOREIGN KEY (date) REFERENCES
Performance
        ON DELETE CASCADE
        ON UPDATE CASCADE)

CREATE TABLE Meet(
        resnum INTEGER,
        name CHAR(20)
        PRIMARY KEY (resnum , name),
        FOREIGN KEY (resnum) REFERENCES
VIP
        ON DELETE CASCADE
        ON UPDATE CASCADE,
        FOREIGN KEY (name) REFERENCES
Performer
        ON DELETE CASCADE
        ON UPDATE CASCADE)
```

Populated Tables:

SeatingSection

sectnum	price	**resnum**
100	140	35142
300	70	49532
203	110	52341
101	150	21435
100	140	31524
209	100	79384
222	100	92448
102	135	41245
300	70	21433
100	140	35193

Watch

resnum	pname	date
35142	Ping	24/04/2022
49532	Mouse Rat	15/08/2024
52341	Scrantonicity	30/06/2022
21435	Duke Silver	14/02/2022
31524	DJ Disco	31/10/2021
79384	Duke Silver	14/02/2022
92448	Mouse Rat	15/08/2024
41245	Ping	29/09/2023
21433	Scrantonicity	30/06/2022
35193	DJ Disco	31/10/2021

GeneralAdmission

resnum
35142
49532
52341
79384
41245

VIP

resnum
21435
31524
92448
21433
35193

Attendee_Seat_1

resnum	cardnum
35142	1234567890123456
49532	1324567890123456
52341	1423567890123456
21435	1523467890123456
31524	1623457890123456
79384	1723456890123456
92448	1823456790123456
41245	1923456780123456
21433	1023456789123456
35193	2134567890123456

Attendee_Seat_2

name	cardnum	**sectnum**	seatnum
Andy Dwyer	1234567890123456	100	2
April Ludgate	1324567890123456	300	3
Ron Swanson	1423567890123456	203	14
Leslie Knope	1523467890123456	101	76
Tom Haverford	1623457890123456	100	57
Jerry Gergich	1723456890123456	209	3
Ann Perkins	1823456790123456	222	35
Ben Wyatt	1923456780123456	102	32
Chris Traeger	1023456789123456	300	14
Donna Meagle	2134567890123456	100	3

Staff

<u>staffID</u>	firstname	lastname
1001	Michael	Scott
1002	Kevin	Malone
1003	Jim	Halpert
1004	Pam	Beesly
1005	Merideth	Palmer
1006	Stanley	Hudson
1007	Dwight	Schrute
1008	Angela	Martin
1009	Oscar	Martinez
1010	Phyllis	Vance
1011	Kelly	Kapur
1012	Ryan	Howard
1013	Toby	Flenderson
1014	Daryl	Philbin
1015	Jan	Levinson

Volunteer

<u>staffID</u>
1001
1005
1009
1010
1011

Technician

<u>staffID</u>	specialization
1002	lighting
1004	sound
1006	lighting
1008	sound
1013	electrical

Security

<u>staffID</u>	company
1003	Securitee
1007	Securitee
1012	SecureRUs
1014	SecureRUs
1015	SoSecure

EntertainmentCompany

name
Entertainment 720
Big Records
Sick Beats
Hype Tunez
XL Entertainment

Performer

name	**cname**
Duke Silver	Entertainment 720
Scrantonicity	Big Records
Mouse Rat	Hype Tunez
Ping	XL Entertainment
DJ Disco	Sick Beats

Travel_Entourage

ename	relationship	**name**
Diane Lewis	wife	Duke Silver
Jon Swanson	son	Duke Silver
Ting	friend	Ping
King	friend	Ping
Crystahl	girlfriend	DJ Disco

Performance 1

date	name	title
13/02/2022	Duke Silver	Smooth as Silver
14/02/2022	Duke Silver	Smooth as Silver
14/02/2022	Ping	Hahaha2
24/04/2022	Ping	Hahaha
29/09/2023	Ping	Hahaha2
30/06/2022	Scrantonicity	Reunion Show
15/08/2024	Mouse Rat	Mouse Rat
30/10/2021	DJ Disco	DJ Disco's Halloween
31/10/2021	DJ Disco	DJ Disco's Halloween

Performance 2

title	type	duration
Smooth as Silver	jazz	120
Hahaha	comedy	90
Hahaha2	comedy	90
Reunion Show	rock	120
Mouse Rat	rock	150
DJ Disco's Hallowe	EDM	180

In_Charge_Of

staffID	**sectnum**	shift
1001	100	13/02/2022
1005	200	13/02/2022
1009	300	15/08/2024
1010	203	30/10/2021
1011	101	30/10/2021
1011	300	31/10/2021

Protect

staffID	**protect-name**
1003	Duke Silver
1007	Scrantonicity
1012	Mouse Rat
1014	Ping
1015	DJ Disco

Meet

resnum	**name**
21435	Duke Silver
31524	DJ Disco
92448	Mouse Rat
21433	Scrantonicity
35193	DJ Disco

Proposed Queries for our Application:

Insertion: Add a performer to the venue’s customer list. Add a performance to the venue’s booking list. Add an attendee to the performer’s meeting list. Add a staff to the venue’s staff list. Add an attendee to a seating section.
Deletion: Remove a performance from the venue’s list of performances. Remove a Performer from the venue’s list of customers. Remove an attendee from the performance’s meeting list. Remove a staff from the venue’s staff list. Remove an attendee from a seating section.
Update: Change performance information for a performance. Change the reservation detail for an attendee (section number, seat number, price). Change the staff information for a staff member. Change entourage details for a given performer. Change information for a performer.
Selection: Search for specific performance meeting desired conditions (date, type, title, duration).
Projection: Choose to see the selected field of all performances (date, type, title, duration).
Join: Join the Attendee and Performance tables to find the name and reservation numbers of all attendees for a specific performance.
Aggregation with Group By: List the number of performances booked (aggregation count) for each performer (group by performer). List the number of performances watched (aggregation count) by each attendee (group by attendee).
Aggregation with Having: List the performances where the number of attendees (aggregation count) is greater than a certain integer (having greater than integer).
Nested Aggregation with Group By: Find the maximum price (aggregation max) among average price of ticket (nested aggregation) for each seating section (group by seating section).
Division: Find all security who protected all the performers.