University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 2

Date: 2023-10-18

Group Number: 145

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Misha Reyes	90302837	q4m4l	mishaoreyes@gmail.com
Nyan Wun Nyunt	86180619	w0k1h	nyannyunt1234@gmail.com
Benedict Damian Tan	61775664	k5v6y	bendamian2012@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.)

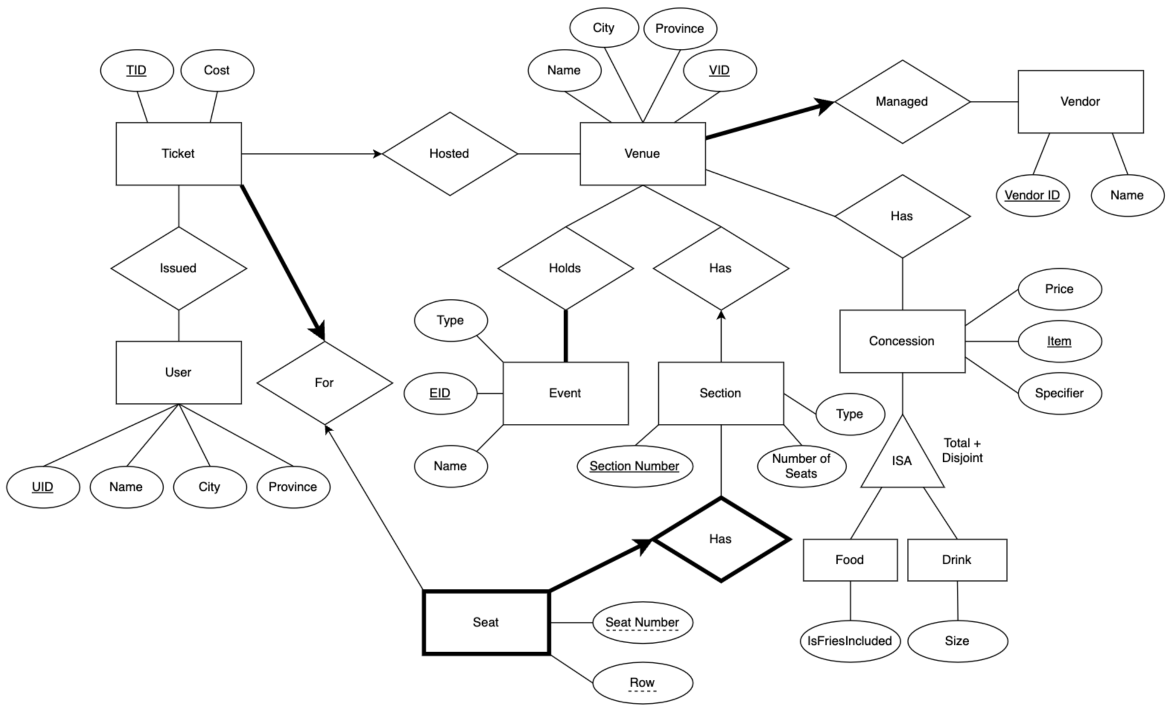
In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

University of British Columbia, Vancouver
Department of Computer Science

2. A brief (~2-3 sentences) summary of your project. Many of your TAs are managing multiple projects so this will help them remember details about your project.

We are creating a ticketing platform that takes care of everything from end to end. It allows venues to create seat maps and issue tickets for it, allows users to buy and resell those tickets, and allows concessions to be packaged with tickets as well. The system is meant to be used in anything from a small fundraising gala to selling tickets for Taylor Swift's Eras tour.

3. The ER diagram you are basing your item #3 (below) on. This ER diagram may be the same as your milestone 1 submission or it might be different. If you have made changes from the version submitted in milestone 1, attach a note indicating what changes have been made and why. If you have decided not to implement the suggestions given by your project mentor, please be sure to leave a note stating why. This is not to say that you must do everything that your project mentor says. In many instances, there are trade-offs between design choices and your decision may be influenced by different factors. That being said, your TAs will often leave suggestions that are meant to help massage your project into a form that will fit with the requirements in future project milestones. If you choose not to take their advice, it would be helpful for them to know why in order to better assist the group moving forward.



4. The schema derived from your ER diagram (above). For the translation of the ER diagram to the relational model, follow the same instructions as in your lectures. The process should be reasonably straightforward. For each table: a. List the table definition (e.g., Table1(attr1: domain1, attr2: domain2, ...)). Make sure to include the domains for each attribute. b. Specify the primary key (PK), candidate key, (CK) foreign keys (FK), and other constraints (e.g., not null, unique, etc.) that the table must maintain.

```
Vendor(
        vendorID: int,
        name: char[128],
        PRIMARY KEY (vendorID)
)
Venue(
        venueID: int,
        vendorID: int NOT NULL,
        name: char[128],
        city: char[128],
        province: char[128]
        PRIMARY KEY (venueID),
        FOREIGN KEY (vendorID) REFERENCES
                Vendor
                        ON DELETE NO ACTION
                        ON UPDATE CASCADE
        UNIQUE (name, address)
)

Concession(
        itemName: char[128],
        price: int,
        specifier: char[1],
        PRIMARY KEY (itemName),
)
```

NOTE:
Specifier here represents a choice of characters (S,M,L,Y,N) that specify **either** isFriesIncluded with Y and N being Yes and No, **or** size with S, M and L being small, medium and large.

University of British Columbia, Vancouver
Department of Computer Science

```
Event(
        eventID: int,
        type: char[5],
        name: char[128],
        venueID: int,
        PRIMARY KEY (eventID)
        FOREIGN KEY (venueID) REFERENCES
                Venue
)

ConcessionVenue(
         itemName: char[128],
         venueID: int,
         PRIMARY KEY (itemName, venueID),
         FOREIGN KEY (itemName)
           REFERENCES   Concessions,
         FOREIGN KEY (venueID)
           REFERENCES Venue
        )

EventVenue(
        eventID: int,
        venueID: int,
        startDateTime: int,
         PRIMARY KEY (eventID, venueID, startDateTime),
        FOREIGN KEY (eventID) REFERENCES Event,
        FOREIGN KEY (venueID) REFERENCES Venues
)

Section(
        numberOfSeats: int,
        type: char[5],
        sectionNumber: int,
        eventID: int,
        venueID: int,
        PRIMARY KEY (eventID, sectionNumber),
        FOREIGN KEY (eventID) REFERENCES
                Event
        FOREIGN KEY (venueID) REFERENCES
                Venue
)
```

University of British Columbia, Vancouver
Department of Computer Science

```
Seat(
        seatNumber: char[3],
        sectionNumber: int,
        eventID: int,
        PRIMARY KEY (sectionNumber, seatNumber),
        FOREIGN KEY (sectionNumber, eventID) REFERENCES
                Section,
                ON DELETE CASCADE
)

User(
        userID: int,
        name: char[128],
        city: char[128],
        province: char[128],
        PRIMARY KEY (userID)
)

Ticket(
        ticketID: int,
        cost: int,
        userID: int,
        venueID: int,
        eventID: int,
        sectionNumber: int NOT NULL,
        seatNumber: int,
        row: char(1) NOT NULL
        PRIMARY KEY (ticketID)
        FOREIGN KEY (venueID) REFERENCES
                Venue
        FOREIGN KEY (eventID) REFERENCES
                Event
        FOREIGN KEY ( sectionNumber) REFERENCES
                Section
        FOREIGN KEY (row, sectionNumber, seatNumber) REFERENCES
                Seat
                ON DELETE NO ACTION
                ON UPDATE CASCADE
        UNIQUE (seatNumber, sectionNumber, eventID, venueID)
```

)

UserTicket(
    userID: int,
    ticketID: int,
    PRIMARY KEY (userID, **ticketID)**
    FOREIGN KEY (userID) REFERENCES
        User
    FOREIGN KEY (ticketID) REFERENCES
        Ticket
)

5. Functional Dependencies (FDs) a. Identify the functional dependencies in your relations, including the ones involving all candidate keys (including the primary key). PKs and CKs are considered functional dependencies and should be included in the list of FDs. You do not need to include trivial FDs such as A → A. Note: In your list of FDs, there must be some kind of valid FD other than those identified by a PK or CK. If you observe that no relations have FDs other than the PK and CK(s), then you will have to intentionally add some (meaningful) attributes to show valid FDs. We want you to get a good normalization exercise. Your design must go through a normalization process.

Vendor(<u>vendorID</u>, name):
    vendorID -> name

Venue(<u>venueID</u>, **vendorID**, name, city, province):
    venueID -> name
    venueID -> city
    venueID -> province
    city -> province

Concession(price, <u>item</u>, specifier, **venueID**):
    item -> specifier
    specifier, item -> price

Event(<u>eventID</u>, type, name, **venueID**):
    eventID -> type
    eventID -> name

Section(numberOfSeats, type, <u>sectionNumber</u>, **eventID**):
    sectionNumber -> type
    sectionNumber -> numberOfSeats
    type -> numberOfSeats

User(<u>userID</u>, name, city, province):

userID -> name
userID -> city
userID -> province
city -> province

Ticket(<u>ticketID</u>, cost, **userID**, **venueID**, **sectionNumber**, **seatNumber**, **row**):

ticketID -> cost
sectionNumber -> cost

6. Normalization a. Normalize each of your tables to be in 3NF or BCNF. Give the list of tables, their primary keys, their candidate keys, and their foreign keys after normalization. You should show the steps taken for the decomposition. Should there be errors, and no work is shown, no partial credit can be awarded without steps shown. The format should be the same as Step 3, with tables listed similar to Table1(attr1:domain1, attr2:domain2, …). ALL Tables must be listed, not only the ones post normalization.

User:

- city -> province violates BCNF, so we decompose on this
  - User1(userID, name, city) User2(city, province)

Section:

- type -> numberOfSeats violates BCNF, so we decompose on this
  - Section1(type, sectionNumber, eventID) Section2(numberOfSeats, type)

Venue:

- city -> province violates BCNF, so we decompose on this
  - Venue1(venueID, vendorID, name, city) Venue2(city, province)

7. The SQL DDL statements required to create all the tables from item #6. The statements should use the appropriate foreign keys, primary keys, UNIQUE constraints, etc. Unless you know that you will always have exactly x characters for a given character, it is better to use the VARCHAR data type as opposed to a CHAR(Y). For example, UBC courses always use four characters to represent which department offers a course. In that case, you will want to use CHAR(4) for the department attribute in your SQL DDL statement. If you are trying to represent the name of a UBC course, you will want to use VARCHAR as the number of characters in a course name can vary greatly.

```
CREATE TABLE Venue1
(venueID INTEGER PRIMARY KEY,
 vendorID INTEGER,
 name VARCHAR(30),
 city VARCHAR(30),
 FOREIGN KEY (city) REFERENCES Venue2(city),
 FOREIGN KEY (vendorID) REFERENCES Vendor(vendorID)
)

CREATE TABLE Venue2
(city VARCHAR(30) PRIMARY KEY,
 province VARCHAR(30)
)


CREATE TABLE Section1
(sectionNumber INTEGER PRIMARY KEY,
 eventID INTEGER,
 type CHAR(5),
FOREIGN KEY (type) REFERENCES Section2(type)
)

CREATE TABLE Section2
(type CHAR(5) PRIMARY KEY,
 numberOfSeats INTEGER
)


CREATE TABLE User1
(userID INTEGER PRIMARY KEY,
 name VARCHAR(30),
 city VARCHAR(30),
 FOREIGN KEY (city) REFERENCES User2(city)
```

```
)

CREATE TABLE User2
(city VARCHAR(30) PRIMARY KEY,
 province VARCHAR(30)
)

CREATE TABLE Vendor
(vendorID INTEGER PRIMARY KEY,
 name VARCHAR(30)
)

CREATE TABLE Concession
(itemName VARCHAR(30) PRIMARY KEY,
 price DECIMAL(10, 2),
 specifier CHAR(1)
)

CREATE TABLE Event
(eventID INTEGER PRIMARY KEY,
 type VARCHAR(30),
 name VARCHAR(30),
 venueID INTEGER,
 FOREIGN KEY (venueID) REFERENCES Venue1(venueID)
)



CREATE TABLE EventVenue
(eventID: INTEGER,
 venueID: INTEGER,
 PRIMARY KEY (eventID, venueID),
 FOREIGN KEY (eventID) REFERENCES Event(eventID),
 FOREIGN KEY (venueID) REFERENCES Venue1(venueID)
)



CREATE TABLE ConcessionVenue
(itemName: VARCHAR(30),
 venueID: INTEGER,
 PRIMARY KEY (itemName, venueID),
 FOREIGN KEY (itemName) REFERENCES Concession(itemName),
 FOREIGN KEY (venueID) REFERENCES Venue1(venueID)
)

CREATE TABLE Section_Seat
(sectionNumber INTEGER,
 seatNumber INTEGER,
PRIMARY KEY (sectionNumber, seatNumber),
FOREIGN KEY (sectionNumber) REFERENCES Section(sectionNumber),
            ON DELETE CASCADE
)

CREATE TABLE Ticket
(ticketID INTEGER PRIMARY KEY,
 cost DECIMAL(10, 2)
)

CREATE TABLE UserTicket
(ticketID INTEGER,
 userID INTEGER,
 PRIMARY KEY (ticketID, userID),
 FOREIGN KEY (ticketID) REFERENCES Ticket(ticketID),
 FOREIGN KEY (userID) REFERENCES User(userID)
)
```

8. INSERT statements to populate each table with at least 5 tuples. You will likely want to have more than 5 tuples so that you can have meaningful queries later.

```
INSERT INTO Vendor
VALUES
(1, “ROGERS”),
(2, “BC Pavillion Company”),
(3, “UBC”),
(4, “SFU”),
(5, “City of Vancouver”);

INSERT INTO Venue
VALUES
(1, 1, “Rogers Arena”, “Vancouver”, “British Columbia”),
(2, 1, “Rogers Centre”, “Toronto”, “Ontario”),
(3, 2, “BC Place”, “Vancouver”, “British Columbia”),
(4, 3, “Thunderbird Stadium”, “Vancouver”, “British Columbia”),
(5, 4, “SFU Stadium at Terry Fox Field”, “Vancouver”, “British Columbia”),
(6, 5, “Queen Elizabeth Theatre”, “Vancouver”, “British Columbia”),
(7, 5, “Annex Theatre”, “Vancouver”, “British Columbia”),
(8, 5, “Orpheum Theatre”, “Vancouver”, “British Columbia”);

INSERT INTO Concession
VALUES
(“Water”, 500, “S”)
(“Coke”, 700, “M”)
(“Beer”, 1000, “L”)
(“Popcorn”, 1000, “N”)
(“Hotdog”, 1200, “N”)
(“Hamburger”, 1500, “Y”)

INSERT INTO ConcessionVenue
VALUES
(“Coke”, 1),
(“Coke”, 2),
(“Coke”, 3),
(“Coke”, 4),
(“Coke”, 5),
(“Water”, 1)
```

```
(“Water”, 2),
(“Water”, 3),
(“Water”, 4),
(“Water”, 5),
(“Beer”, 1),
(“Beer”, 2),
(“Beer”, 3),
(“Beer”, 4),
(“Beer”, 5),
(“Popcorn”, 1),
(“Popcorn”, 2),
(“Popcorn”, 3),
(“Popcorn”, 4),
(“Popcorn”, 5),
(“Hotdog”, 1),
(“Hotdog”, 2),
(“Hotdog”, 3),
(“Hotdog”, 4),
(“Hotdog”, 5),
(“Hamburger”, 1),
(“Hamburger”, 2),
(“Hamburger”, 3),
(“Hamburger”, 4),
(“Hamburger”, 5);

INSERT INTO Event
VALUES
(1, “CNCRT”, “Taylor Swift Eras Tour”)
(2, “DANCE”, “Halloween Dance”)
(3, “PLAYS”, “The Nutcracker”)
(4, “SPORT”, “Baseball Game”)
(5, “SPORT”, “Hockey Game”)

INSERT INTO EventVenue
VALUES
(1, 1, 1697853600),
(1, 1, 1697940000),
(1, 2, 1698447600),
(4, 2, 1697853600),
(5, 3, 1698447600),
(5, 3, 1697853600),
(5, 4, 1698447600),
(2, 5, 1698447600),
(3, 6, 1698447600),
(3, 7, 1698447600),
(3, 8, 1698447600);

INSERT INTO Section
VALUES
(100, “SEATS”, 1, 1, 1)
(100, “SEATS”, 2, 1, 1)
(100, “SEATS”, 3, 1, 1)
(100, “SEATS”, 4, 1, 1)
(1000, “STAND”, 5, 1, 1)

INSERT INTO Seat
VALUES
(“A01”, 1, 1)
(“A02”, 1, 1)
(“A03”, 1, 1)
(“A04”, 1, 1)
(“A05”, 1, 1)

INSERT INTO Usesr
VALUES
(1, “John Smith”, “Vancouver”, “British Columbia”)
(2, “Joe Flow”, “Vancouver”, “British Columbia”)
(3, “Johnathan Smithanan”, “Vancouver”, “British Columbia”)
(4, “Florance Nightingale”, “Vancouver”, “British Columbia”)
(5, “Evangeline Longbottom”, “Toronto”, “Ontario”)

INSERT INTO Ticket
VALUES
(1, 10000, 1, 1, 1, 1, “A01”, “A”)
(2, 10000, 1, 1, 1, 1, “A02”, “A”)
(3, 10000, 2, 1, 1, 1, “A03”, “A”)
(4, 10000, 3, 1, 1, 1, “A04”, “A”)
(5, 10000, 4, 1, 1, 5, NULL, “A”)
```