CPSC 304 Project Cover Page

Milestone #: 2

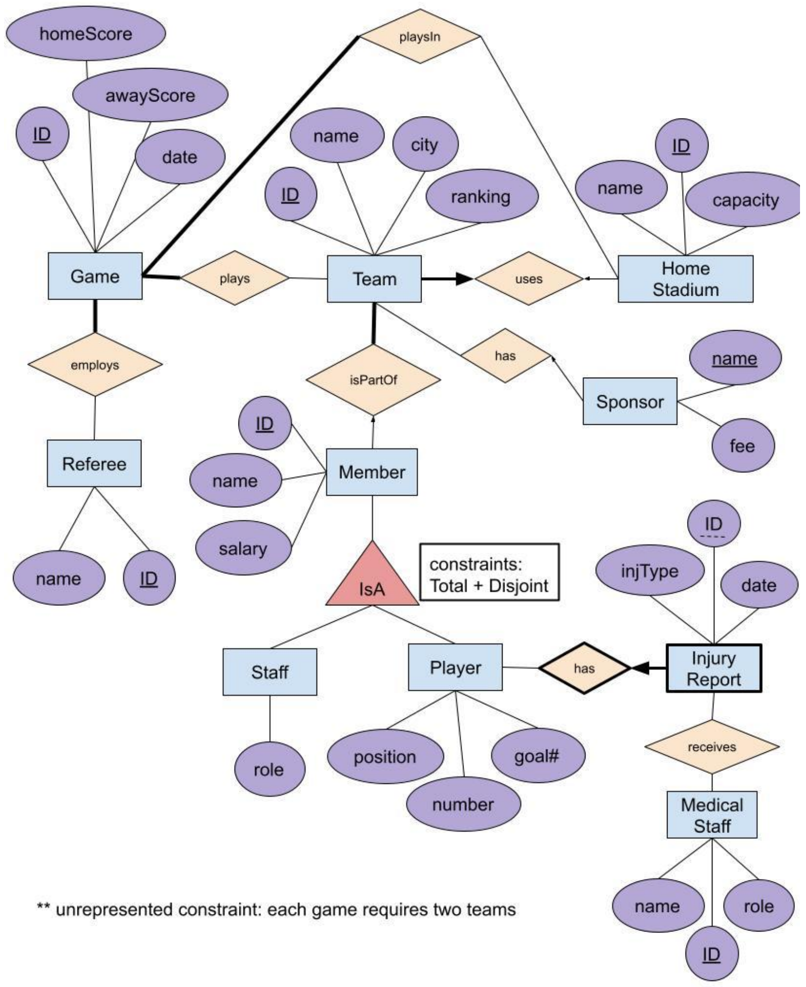
Date: March 1, 2023

Group Number: 91

Name	**Student Number**	**CS Alias (Userid)**	**Preferred E-mail Address**
Seo Woon Baik	65871007	h8w6k	sunnybak@student.ubc.ca
Suyeon Choi	33154717	y8l1w	kraton727@gmail.com
Ryoh Cuahutle	26071787	f6k1b	ryoh.ct@gmail.com

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.
    - Our project is a soccer league management database. You are able to query for teams and the games they will be playing, the players on teams, and injuries sustained by players, etc.

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.



Some changes in our ER-diagram:

- Added an ID attribute in some entities as their primary key for simplicity.

- Splitted the “score” attribute of Game into “homeScore” and “awayScore” to save them in integer.
- Added one more relation (PlaysIn) as we previously only had 6 relations excluding the weak entity.

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.

- Game(<u>gameID</u>: integer, gameDate: date, homeScore: integer, awayScore: integer)
    - PK: gameID
    - CK: gameID
    - FK: none
- Team(<u>teamID</u>: integer, **stadiumID**: integer, teamName: varchar[30], city: varchar[85], ranking: integer)
    - PK: teamID
    - CK: teamID
    - FK: stadiumID references HomeStadium
    - UNIQUE: stadiumID
    - NOT NULL: teamName, city
- HomeStadium(<u>stadiumID</u>: integer, stadName: varchar[20], capacity: integer)
    - PK: stadiumID
    - CK: stadiumID, stadName
    - FK: none
    - UNIQUE: stadName
    - NOT NULL: stadName
- Sponsor(<u>sponsorName</u>: varchar[30], fee: integer, **teamID**: integer)
    - PK: sponsorName
    - CK: sponsorName
    - FK: teamID references Team
- Referee(<u>refereeID</u>: integer, refName: varchar[30])
    - PK: refereeID
    - CK: refereeID
    - FK: none
    - NOT NULL: refName
- Staff(<u>memberID</u>: integer, staffName: varchar[30], staffSalary: integer, role: varchar[20], **teamID**: integer)
    - PK: memberID
    - CK: memberID
    - FK: teamID references Team
    - NOT NULL: staffName

- Player(<u>memberID</u>: integer, playerName: varchar[30], playerSalary: integer, position: varchar[20], goalNum: integer, number: integer, **teamID**: integer)
    - PK: memberID
    - CK: memberID, number
    - FK: teamID references Team
    - UNIQUE: number
    - NOT NULL: playerName
- InjuryReport(<u>injID</u>: integer, <u>**memberID**</u>: integer, injType: varchar[50], date: date)
    - PK: (injID, memberID)
    - CK: (injID, memberID)
    - FK: memberID references Member
- MedicalStaff(<u>mstaffID</u>: integer, mstaffName: varchar[20], mstaffRole: varchar[20])
    - PK: mstaffID
    - CK: mstaffID
    - FK: none
    - NOT NULL: mstaffName
- Employs(<u>**gameID**</u>: integer, <u>**refereeID**</u>: integer)
    - PK: (gameID, refereeID)
    - CK: (gameID, refereeID)
    - FK: gameID references Game, refereeID references Referee
- Plays(<u>**gameID**</u>: integer, <u>**teamID1**</u>: integer, <u>**teamID2**</u>: integer)
    - PK: (gameID, teamID1, teamID2)
    - CK: (gameID, teamID1, teamID2)
    - FK: gameID references Game, (teamID1, teamID2) references Team
- Receives(<u>**mstaffID**</u>: integer, <u>**memberID**</u>: integer, <u>**injID**</u>: integer)
    - PK: (mstaffID, memberID, injID)
    - CK: (mstaffID, memberID, injID)
    - FK: mstaffID references MedicalStaff, memberID references Member, injID references InjuryReport
- PlaysIn(<u>**gameID**</u>: integer, <u>**stadiumID**</u>: integer)
    - PK: (gameID, stadiumID)
    - CK: (gameID, stadiumID)
    - FK: gameID references Game, stadiumID references HomeStadium

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.

Game:
- gameID -> gameDate, homeScore, awayScore

Team:
- teamID -> stadiumID, teamName, city, ranking
- ranking -> teamName

HomeStadium:
- stadiumID -> stadName, capacity

Sponsor:
- sponsorName -> fee, teamID

Referee:
- refereeID -> refName

Staff:
- memberID -> staffName, staffSalary, role, teamID

Player:
- memberID -> playerName, playerSalary, position, goalNum, number, teamID
- playerName, number, teamID -> playerSalary, position, goalNum

InjuryReport:
- memberID, injID -> injType, date

MedicalStaff:
- mstaffID -> mstaffName, mstaffRole

Employs:
- No non-trivial FDs

Plays:
- No non-trivial FDs

Receives:
- No non-trivial FDs

PlaysIn:
- No non-trivial FDs

** We've added the highlighted FDs for the normalization process

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.

Game(<u>gameID</u>, gameDate, homeScore, awayScore):

- (<u>gameID</u>) -> gameDate, homeScore, awayScore
- Already in BCNF and 3NF

Team(<u>teamID</u>, **stadiumID**, teamName, city, ranking):

(1) (<u>teamID</u>) -> stadiumID, teamName, city, ranking
(2) (ranking) -> teamName

- FD (2) violates BCNF and 3NF (ranking is not a superkey, teamName is not part of key)

  Decompose:

  R1(<u>ranking</u>, teamName), R2(ranking, <u>teamID</u>, stadiumID, city)

  - ranking -> teamName holds for R1, so R1 is in BCNF.
  - teamID -> ranking, stadiumID, city holds for R2, so R2 is also in BCNF.

HomeStadium(<u>stadiumID</u>, stadName, capacity):

- (<u>stadiumID</u>) -> stadName, capacity
- Already in BCNF and 3NF

Sponsor(<u>sponsorName</u>, fee, **teamID**):

- (<u>sponsorName</u>) -> fee, teamID
- Already in BCNF and 3NF

Referee(<u>refereeID</u>, refName):

- (<u>refereeID</u>) -> refName
- Already in BCNF and 3NF

Staff(<u>memberID</u>, staffName, staffSalary, role, **teamID**):

- (<u>memberID</u>) -> staffName, staffSalary, role, teamID
- Already in BCNF and 3NF

Player(memberID, playerName, playerSalary, position, goalNum, number, **teamID**):

(1) (memberID) -> playerName, playerSalary, position, goalNum, number, teamID
(2) (playerName, number, teamID) -> playerSalary, position, goalNum

- FD (2) violates BCNF and 3NF (LHS is not a superkey; RHS is not part of key)

  Decompose:

  R1(playerName, number, teamID, playerSalary, position, goalNum), R2(playerName, number, teamID, memberID)

  - playerName, number, teamID -> playerSalary, position, goalNum holds for R1, so R1 is in BCNF.
  - memberID -> playerName, number, teamID holds for R2, so R2 is also in BCNF.

InjuryReport(injID, **memberID**, injType, date):
- (memberID, injID) -> injType, date
- Already in BCNF and 3NF

MedicalStaff(mstaffID, mstaffName, mstaffRole):
- (mstaffID) -> mstaffName, mstaffRole
- Already in BCNF and 3NF

Employs(**gameID**, **refereeID**):
- No non-trivial FDs

Plays(**gameID**, **teamID1**, **teamID2**):
- No non-trivial FDs

Receives(**mstaffID**, **memberID**, **injID**):
- No non-trivial FDs

PlaysIn(**gameID**, **stadiumID**)
- No non-trivial FDs

7. The SQL DDL statements required to create all the tables from item #4(5?). The statements should use the appropriate foreign keys, primary keys, UNIQUE constraints, etc.

```
CREATE TABLE Game (
        gameID int PRIMARY KEY,
        gameDate Date,
        homeScore int,
        awayScore int
        )

CREATE TABLE Team (
        teamID int PRIMARY KEY,
        stadiumID int UNIQUE,
        teamName char[30] NOT NULL,
        city char[30] NOT NULL,
        ranking int,
        FOREIGN KEY (stadiumID) REFERENCES HomeStadium ON DELETE SET NULL
        )

CREATE TABLE HomeStadium (
        stadiumID int PRIMARY KEY,
        name char[30] NOT NULL,
        capacity int
)

CREATE TABLE Sponsor (
        sponsorName char[30] PRIMARY KEY,
        fee int,
        teamID int,
        FOREIGN KEY (teamID) REFERENCES Team ON DELETE SET NULL
)

CREATE TABLE Referee (
        refereeID int PRIMARY KEY,
        refName char[30] NOT NULL
)

CREATE TABLE Staff (
        memberID int PRIMARY KEY,
        staffName char[30] NOT NULL,
        staffSalary int,
        role char[20],
        teamID int,
        FOREIGN KEY (teamID) REFERENCES Team ON DELETE SET NULL
)
```

```
CREATE TABLE Player (
	memberID int PRIMARY KEY,
	playerName char[30] NOT NULL,
	playerSalary int,
	position char[20],
	goalNum int,
	number int UNIQUE,
	teamID int,
	FOREIGN KEY (teamID) REFERENCES Team ON DELETE SET NULL
)

CREATE TABLE Injury Report (
	injID int,
	memberID int,
	injType char[50],
	date Date,
	PRIMARY KEY(injID, memberID),
	FOREIGN KEY (memberID) REFERENCES Player ON DELETE CASCADE
)

CREATE TABLE Medical Staff (
	mstaffID int PRIMARY KEY,
	mstaffName char[20] NOT NULL,
	mstaffRole char[20]
)

CREATE TABLE Employs (
	gameID int,
	refereeID int,
	PRIMARY KEY(gameID, refereeID),
	FOREIGN KEY (gameID) REFERENCES Game ON DELETE CASCADE,
	FOREIGN KEY(refereeID) REFERENCES Referee ON DELETE CASCADE
)

CREATE TABLE Plays (
	gameID int,
	teamID1 int,
	teamID2 int,
	PRIMARY KEY(gameID, teamID1, teamID2),
	FOREIGN KEY (gameID) REFERENCES Game ON DELETE CASCADE,
	FOREIGN KEY (teamID1, teamID2) REFERENCES Team ON DELETE CASCADE
)

CREATE TABLE Receives (
	mstaffID int,
	memberID int
	injID int,
	PRIMARY KEY(mstaffID, memberID, injID),
	FOREIGN KEY(mstaffID) REFERENCES Medical Staff ON DELETE CASCADE,
	FOREIGN KEY(memberID) REFERENCES Member ON DELETE CASCADE,
	FOREIGN KEY(injID) REFERENCES InjuryReport ON DELETE CASCADE
)

CREATE TABLE PlaysIn (
	gameID int,
	stadiumID int,
	PRIMARY KEY(gameID, stadiumID),
	FOREIGN KEY (gameID) REFERENCES Game ON DELETE CASCADE,
	FOREIGN KEY (stadiumID) REFERENCES HomeStadium ON DELETE CASCADE
)
```

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 on.

```
INSERT INTO Game (gameID, gameDate, homeScore, awayScore) VALUES
(1, '2022-01-01', 2, 1),
(2, '2022-01-05', 0, 0),
(3, '2022-01-12', 3, 2),
(4, '2022-01-20', 1, 1),
(5, '2022-01-27', 2, 3);

INSERT INTO HomeStadium (stadiumID, name, capacity) VALUES
(1, 'Old Trafford', 74879),
(2, 'Etihad Stadium', 55017),
(3, 'Anfield', 53394),
(4, 'Emirates Stadium', 60260),
(5, 'Stamford Bridge', 40853);

INSERT INTO Team (teamID, stadiumID, teamName, city, ranking) VALUES
(1, 1, 'Manchester United', 'Manchester', 2),
(2, 2, 'Manchester City', 'Manchester', 1),
(3, 3, 'Liverpool', 'Liverpool', 4),
(4, 4, 'Arsenal', 'London', 8),
(5, 5, 'Chelsea', 'London', 3);

INSERT INTO Sponsor (sponsorName, fee, teamID) VALUES
('Coca-Cola', 1000000, 1),
('Nike', 1500000, 2),
('Pepsi', 800000, 3),
('Adidas', 1200000, 4),
('Samsung', 900000, 5);
```

```
INSERT INTO Referee (refereeID, refName) VALUES
(1, 'John Smith'),
(2, 'Mary Johnson'),
(3, 'David Lee'),
(4, 'Emily Wong'),
(5, 'Michael Chen');

INSERT INTO Staff (memberID, staffName, staffSalary, role, teamID) VALUES
(6, 'Tom Lee', 50000, 'Manager', 1),
(7, 'Grace Kim', 40000, 'Coach', 1),
(8, 'Emma Liu', 35000, 'Trainer', 2),
(9, 'Kevin Chen', 60000, 'Manager', 3),
(10, 'Sophia Wang', 45000, 'Coach', 4);

INSERT INTO Player (memberID, playerName, playerSalary, position, goalNum, number,
teamID) VALUES
(1, 'David Beckham', 1000000, 'Forward', 10, 7, 1),
(2, 'Cristiano Ronaldo', 2000000, 'Forward', 25, 10, 1),
(3, 'Lionel Messi', 2500000, 'Forward', 30, 10, 2),
(4, 'Neymar Jr', 1800000, 'Forward', 20, 10, 3),
(5, 'Kylian Mbappe', 1500000, 'Forward', 15, 7, 4);

INSERT INTO InjuryReport (injID, memberID, injType, date) VALUES
(1, 1, 'Ankle Sprain', '2022-02-10'),
(2, 3, 'Hamstring Strain', '2022-02-15'),
(3, 5, 'Knee Injury', '2022-02-17'),
(4, 2, 'Concussion', '2022-02-20'),
(5, 4, 'Groin Strain', '2022-02-25');

INSERT INTO Medical Staff (mstaffID, mstaffName, mstaffRole) VALUES
(1, 'Dr. James Lee', 'Physician'),
(2, 'Dr. Susan Kim', 'Physiotherapist'),
(3, 'Dr. David Park', 'Chiropractor'),
(4, 'Dr. Emily Chen', 'Massage Therapist'),
(5, 'Dr. Michael Wang', 'Athletic Trainer');
```

```
INSERT INTO Employs (gameID, refereeID) VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5);

INSERT INTO Plays (gameID, teamID1, teamID2) VALUES
(1, 1, 2),
(2, 3, 4),
(3, 5, 1),
(4, 2, 3),
(5, 4, 5);

INSERT INTO Receives (mstaffID, memberID, injID) VALUES
(1, 1, 1),
(2, 3, 2),
(3, 5, 3),
(4, 2, 4),
(5, 4, 5);

INSERT INTO PlaysIn (gameID, stadiumID) VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5);
```