# CPSC 304 Project Cover Page

Milestone #: 2

Date: Oct. 15, 2024

Group Number: 84

| **Name** | **Student Number** | **CS Alias (Userid)** | **Preferred E-mail Address** |
|---|---|---|---|
| Connor Won | 27314798 | c6c6o | cywon2@gmail.com |
| Jonathan Cai | 37435609 | g3f5e | jonathan.y.cai@outlook.com |
| Harry Zhu | 98024730 | t8i3b | harrylakers242424@gmail.com |

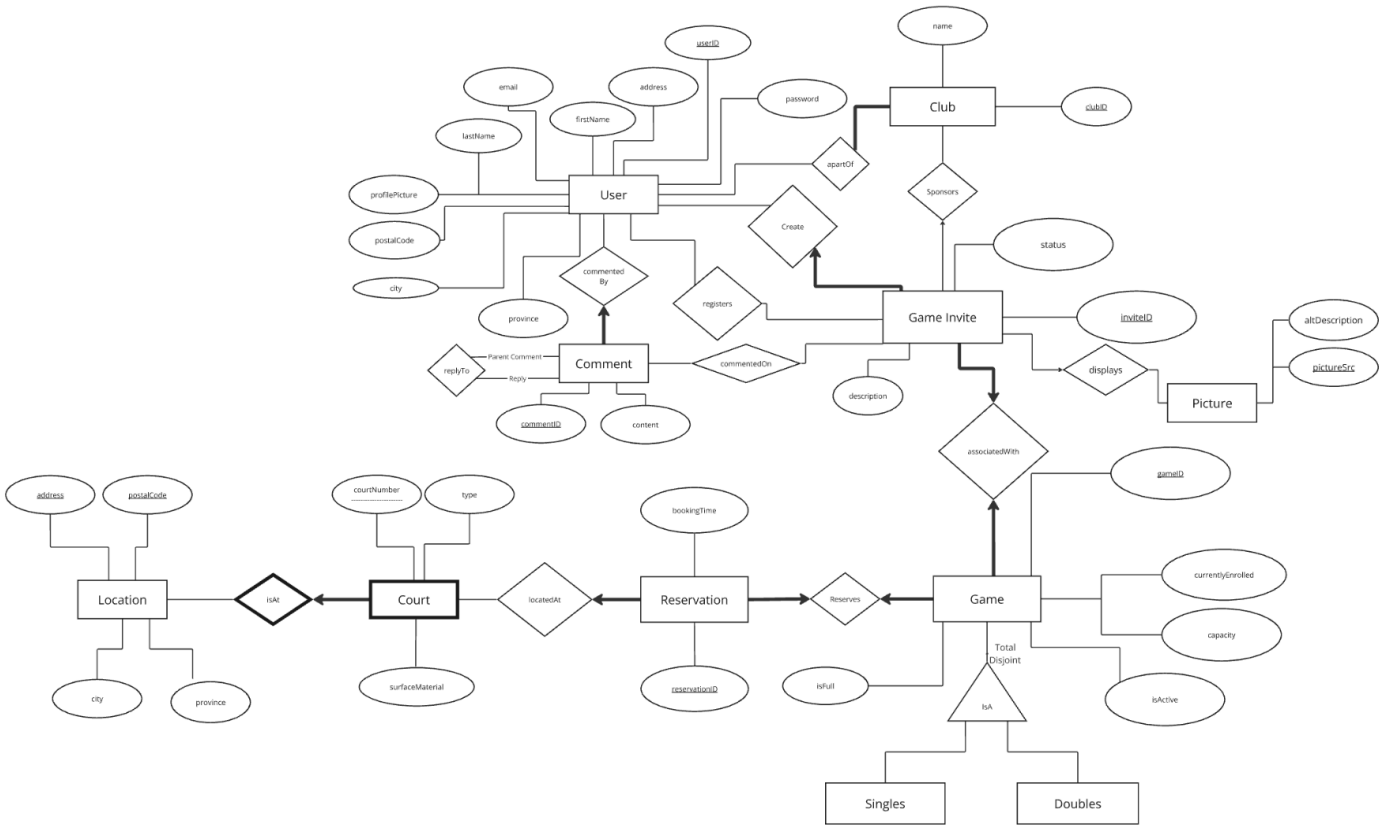
By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above.

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

# SUMMARY OF PROJECT (~ 2-3 SENTENCES)

The domain of this application is sports and recreation. We will be creating an application where pickleball players can connect and play with one another at their local courts. With a simple click of a button, they can register/ host games, join clubs and make friends and memories.

# ER DIAGRAM




Changes from Milestone 1:

- We decided to implement the changes suggested by our project mentor and removed the aggregated relationship between court and location. Instead we decided to maintain the weak entity relationship and made relationships directly with court
- We added an additional Reservation entity as we realized with the old structure, each Court could only be associated with a single game. However, how it should work is that each court should be able to be associated with multiple games, but at a specific time it can only be associated with a single game. So, the introduction of this new entity solves this issue
- The relationship between GameInvite and Picture is now a many-to-one relationship as each GameInvite can only display one image
- Location received *city* and *province* attributes to further clarify its location
- Court received *surfaceMaterial* attribute to differentiate surface type so users can better prepare for their games
- Game received *isFull* attribute to better visualize when to turn off the GameInvite
- User received *postalCode*, *province,* and *city* to improve filtering for GameInvites

# SCHEMA

Location(
    address: char PK,
    postalCode: char PK,
    city: char,
    province: char,
    PK(address, postalCode)
)

Court(
    courtNumber: integer PK,
    type: char NOT NULL,
    **address**: char PK,
    **postalCode**: char PK,
    surfaceMaterial: char NOT NULL,
    FK(address, postalCode) REF Location(address, postalCode),
    PK(courtNumber, address, postalCode),
)

Reservation(
    reservationID: integer PK,
    bookingTime: date NOT NULL,
    **courtNumber**: integer NOT NULL,
    **address**: char NOT NULL,
    **postalCode**:char NOT NULL,
    CK(bookingTime, courtNumber, address, postalCode)
    FK(courtNumber, address, postalCode) REF Court(courtNumber, address, postalCode),
)

Singles(
    gameID: integer PK,
    **reservationID**: integer UNIQUE NOT NULL,
    **gameInviteID**: integer UNIQUE NOT NULL,
    currentlyEnrolled: integer NOT NULL,
    capacity: integer NOT NULL,
    isActive: boolean NOT NULL,
    isFull: boolean NOT NULL,
    CK(reservationID),
    CK(gameInviteID),
    FK(reservationID) REF Reservation,
    FK(gameInviteID) REF GameInvite

)

Doubles(
    gameID: integer PK,
    **reservationID**: integer UNIQUE NOT NULL,
    **gameInviteID**: integer UNIQUE NOT NULL,
    currentlyEnrolled: integer NOT NULL,
    capacity: integer NOT NULL,
    isActive: boolean NOT NULL,
    isFull: boolean NOT NULL,
    CK(reservationID),
    CK(gameInviteID),
    FK(reservationID) REF Reservation,
    FK(gameInviteID) REF GameInvite
)

Picture(
    pictureSrc: char PK,
    altDescription: char
)

GameInvite(
    inviteID: integer PK,
    status: boolean NOT NULL,
    description: char,
    **creator**: integer NOT NULL,
    **sponsor**: integer,
    **thumbnail**: char,
    FK(creator) REF User,
    FK(sponsor) REF Club,
    FK(thumbnail) REF Picture,
)

Club(
    clubID: integer PK,
    name: char NOT NULL,
)

User(
    userID: integer PK,
    email: char UNIQUE NOT NULL,
    password: char NOT NULL,
    firstName: char NOT NULL,

lastName: char NOT NULL,
profilePicture: char,
address: char NOT NULL,
postalCode: char NOT NULL,
province: char NOT NULL,
city: char NOT NULL,
CK(email),
)

Comment(
commentID: integer PK,
content: char NOT NULL,
**commentedBy**: integer NOT NULL,
FK(commentedBy) REF User,
)

replyTo(
**parentID**: integer PK,
**replyID**: integer PK,
PK(parentID, replyID),
FK(parentID) REF Comment,
FK(replyID) REF Comment
)

commentedOn(
**commentID**: integer PK,
**inviteID**: integer PK,
PK(commentID, inviteID),
FK(commentID) REF Comment,
FK(inviteID) REF GameInvite
)

apartOf(
**userID**: integer PK,
**clubID**: integer PK,
PK(userID, clubID),
FK(userID) REF User,
FK(clubID) REF Club
)

registers(
**userID**: integer PK,

**inviteID**: integer PK,
PK(userID, inviteID),
FK(userID) REF User,
FK(inviteID) REF GameInvite
)

Note:

- After analyzing the initial schema, we added the following attributes to create more functional dependencies for normalization, but we also wanted to better model the functionalities we plan on implementing.
    - Location: Added *city* and *province*
    - Court: Added *surfaceMaterial*
    - Game: Added *isFull*
    - User: Added *postalCode, province* and *city*

# FUNCTIONAL DEPENDENCIES

Location FD's:

- address, postalCode → city, province
- postalCode → city, province

Court FD's:

- address, postalCode, courtNumber → type, surfaceMaterial
- surfaceMaterial → type

Reservation FD's:

- reservationID → bookingTime, courtNumber, address, postalCode
- bookingTime, courtNumber, address, postalCode → reservationID

Singles FD's:

- gameID → reservationID, gameInviteID, currentlyEnrolled, capacity, isActive, isFull
- reservationID → gameID, gameInviteID, currentlyEnrolled, capacity, isActive, isFull
- gameInviteID → gameID, reservationID, currentlyEnrolled, capacity, isActive, isFull
- currentlyEnrolled, capacity → isFull

Doubles FD's:

- gameID → reservationID, gameInviteID, currentlyEnrolled, capacity, isActive, isFull
- reservationID → gameID, gameInviteID, currentlyEnrolled, capacity, isActive, isFull
- gameInviteID → gameID, reservationID, currentlyEnrolled, capacity, isActive, isFull
- currentlyEnrolled, capacity → isFull

Picture FD's:

- pictureSrc → altDescription

GameInvite FD's:

- inviteID → status, description, creator, sponsors, thumbnail, gameID

Club FD's:

- clubID → name

User FD's

- userID → password, address, firstName, lastName, email, profilePicture, postalCode, province, city
- email → userID, password, address, firstName, lastName, profilePicture, postalCode, province, city
- postalCode → province, city
- address, province, city → postalCode

Comment FD's:

- commentID → content, commentedBy

replyTo FD's:

- parentID → replyID
- replyID → parentID

commentedOn FD's:

- commentID → inviteID
- inviteID → commentID

apartOf: FD's

- userID → clubID
- clubID → userID

registers: FD's

- userID → inviteID
- inviteID → userID

# NORMALIZATION

## LOCATION

Give symbolic labels for each attribute to make normalization more readable:

- Address → A,
- PostalCode → PC
- City → C
- Province → P

Now we have the relation: Location(A, PC, C, P)

FDs for this relation:

- A, PC → C, P
- PC → C, P

Closures:

- $(A, PC)^+ = \{A, PC, C, P\}$
- $(PC)^+ = \{PC, C, P\}$

Determine Minimal Key:

| Left | Middle | Right |
|---|---|---|
| A, PC | | C, P |

Minimal Key: (A, PC)

Finding Minimal Covers:

1. Put FDs into standard form:
    - A, PC → C
    - A, PC → P
    - PC → C
    - PC → P
2. Minimize LHS of each FD
    - Nothing to minimize
3. Delete Redundant FD:
    - PC → C
    - PC → P
4. Minimal Cover:
    - PC → C
    - PC → P

Our FDs are not in 3NF, so we will decompose using the Synthesis method:

1. Create relations for each FD:

    - $R_1$(PC, C)
    - $R_2$(PC, P)
2. Add a relation that includes minimal key:
    - $R_3$(A, PC)
3. Optimize Decomposition:
    - No optimizations to be made
4. Relations we are left with:
    - $R_1$(PC, C)
    - $R_2$(PC, P)
    - $R_3$(A, PC)

SCHEMAS:

Location(
    address: char PK,
    **postalCode**: char PK,
    PK(address, postalCode)
    FK(postalCode) REF CityLocation,
)

CityLocation(
    postalCode: char PK,
    city: char NOT NULL,
)

ProvinceLocation(
    **postalCode**: char PK,
    province: char NOT NULL,
    FK(postalCode) REF CityLocation,
)

---

**SINGLES**

Give symbolic labels for each attribute to make normalization more readable:

- gameID → G
- reservationID → R
- gameInviteID → GI
- currentlyEnrolled→ CE
- capacity → CA
- isActive → IA
- isFull → IF

Now we have the relation: Singles(G, R, GI, CE, CA, IA, IF)

FDs for this relation:

- G → R, GI, CE, CA, IA, IF
- R → G, GI, CE, CA, IA, IF
- GI → G, R, CE, CA, IA, IF
- CE, CA → IF

Closures:

- $(G)^+$ = {G, R, GI, CE, CA, IA, IF}
- $(R)^+$ = {R, G, GI, CE, CA, IA, IF}
- $(GI)^+$ = {GI, G, R, CE, CA, IA, IF}
- $(CE, CA)^+$ = {CE, CA, IF}

Determine Minimal Key:

| Left | Middle | Right |
|---|---|---|
| | G, R, GI, CE, CA | IA, IF |

Possible Minimal Keys: G or R or GI

Finding Minimal Covers:

1. Put FDs into standard form:
    - G → R
    - G → GI
    - G → CE
    - G → CA
    - G → IA
    - G → IF
    - R → G
    - R → GI
    - R → CE
    - R → CA
    - R → IA
    - R → IF
    - GI → G
    - GI → R
    - GI → CE
    - GI → CA
    - GI → IA
    - GI → IF
    - CE, CA → IF
2. Minimize LHS of each FD:
    - Nothing to minimize

3. Delete Redundant FD:
    - G → GI
    - R → GI
    - GI → G
    - GI → R
    - GI → CE
    - GI → CA
    - GI → IA
    - CE, CA → IF
4. Minimal Cover:
    - G → GI
    - R → GI
    - GI → G
    - GI → R
    - GI → CE
    - GI → CA
    - GI → IA
    - CE, CA → IF

Our FDs are not in 3NF, so we will decompose using the Lossless Join method:

1. CE, CA → IF violates 3NF, decompose to BCNF:
    - $R_1$(CE, CA, IF)
    - $R_2$(G, R, GI, CE, CA, IA)
2. No other FDs violate BCNF and no FDs are lost so we are left with:
    - $R_1$(CE, C, IF)
    - $R_2$(G, R, GI, CE, CA, IA)

SCHEMAS:

Singles(

| | |
|---|---|
| gameID: integer | PK, |
| **reservationID**: integer | UNIQUE NOT NULL, |
| **gameInviteID**: integer | UNIQUE NOT NULL, |
| **currentlyEnrolled**: integer | NOT NULL, |
| **capacity**: integer | NOT NULL, |
| isActive: boolean | NOT NULL, |
| CK(reservationID), | |
| CK(gameInviteID), | |
| FK(reservationID) REF Reservation, | |
| FK(gameInviteID) REF GameInvite, | |
| FK(currentlyEnrolled, capacity) REF SinglesStatus | |

)

SinglesStatus(

currentlyEnrolled: integer PK,
capacity: integer PK,
isFull: boolean NOT NULL,
PK(currentlyEnrolled, capacity),
)

---

**DOUBLES**

Give symbolic labels for each attribute to make normalization more readable:

- gameID → G
- reservationID → R
- gameInviteID → GI
- currentlyEnrolled→ CE
- capacity → CA
- isActive → IA
- isFull → IF

Now we have the relation: Doubles(G, R, GI, CE, CA, IA, IF)

FDs for this relation:

- G → R, GI, CE, CA, IA, IF
- R → G, GI, CE, CA, IA, IF
- GI → G, R, CE, CA, IA, IF
- CE, CA → IF

Closures:

- $(G)^+$ = {G, R, GI, CE, CA, IA, IF}
- $(R)^+$ = {R, G, GI, CE, CA, IA, IF}
- $(GI)^+$ = {GI, G, R, CE, CA, IA, IF}
- $(CE, CA)^+$ = {CE, CA, IF}

Determine Minimal Key:

| Left | Middle | Right |
|---|---|---|
| | G, R, GI, CE, CA | IA, IF |

Possible Minimal Keys: G or R or GI

Finding Minimal Covers:

1. Put FDs into standard form:
    - G → R
    - G → GI

- G → CE
- G → CA
- G → IA
- G → IF
- R → G
- R → GI
- R → CE
- R → CA
- R → IA
- R → IF
- GI → G
- GI → R
- GI → CE
- GI → CA
- GI → IA
- GI → IF
- CE, CA → IF

2. Minimize LHS of each FD:
    - Nothing to minimize
3. Delete Redundant FD:
    - G → GI
    - R → GI
    - GI → G
    - GI → R
    - GI → CE
    - GI → CA
    - GI → IA
    - CE, CA → IF
4. Minimal Cover:
    - G → GI
    - R → GI
    - GI → G
    - GI → R
    - GI → CE
    - GI → CA
    - GI → IA
    - CE, CA → IF

Our FDs are not in 3NF, so we will decompose using the Lossless Join method:

1. CE, CA → IF violates 3NF, decompose to BCNF:
    - $R_1$(<u>CE</u>, <u>CA</u>, IF)
    - $R_2$(<u>G</u>, R, GI, CE, CA, IA)
2. No other FDs violate BCNF and no FDs are lost so we are left with:
    - $R_1$(<u>CE</u>, <u>C</u>, IF)

- $R_2$(G, R, GI, CE, CA, IA)

SCHEMAS:

Doubles(
    gameID: integer PK,
    **reservationID**: integer UNIQUE NOT NULL,
    **gameInviteID**: integer UNIQUE NOT NULL,
    **currentlyEnrolled**: integer NOT NULL,
    **capacity**: integer NOT NULL,
    isActive: boolean NOT NULL,
    CK(reservationID),
    CK(gameInviteID),
    FK(reservationID) REF Reservation,
    FK(gameInviteID) REF GameInvite,
    FK(currentlyEnrolled, capacity) REF DoublesStatus
)

DoublesStatus(
    currentlyEnrolled: integer PK,
    capacity: integer PK,
    isFull: boolean NOT NULL,
    PK(currentlyEnrolled, capacity),
)

---

## USER

Give symbolic labels for each attribute to make normalization more readable:

- userID → UI
- password → P
- address → A
- firstName → FN
- lastName → LN
- email → E
- profilePicture → PP
- postalCode → PC
- province → PR
- city → C

Now we have the relation: User(UI, P, A, FN, LN, E, PP, PC, PR, C)

FDs for this relation:

- UI → P, A, FN, LN, E, PP, PC, PR, C

- E → UI, P, A, FN, LN, PP, PC, PR, C
- PC → PR, C
- A, PR, C → PC

Closures:

- $(UI)^+ \rightarrow$ {UI, P, A, FN, LN, E, PP, PC, PR, C}
- $(E)^+ \rightarrow$ {E, UI, P, A, FN, LN, PP, PC, PR, C}
- $(PC)^+ \rightarrow$ {PC, PR, C}
- $(A, PR, C)^+ \rightarrow$ {A, PR, C, PC}

Determine Minimal Key:

| Left | Middle | Right |
|---|---|---|
| | UI, E, PC, A, PR, C | P, FN, LN, PP |

Possible Minimal Keys: UI or E

Finding Minimal Covers:

1. Put FDs into standard form:
    - UI → P
    - UI → A
    - UI → FN
    - UI → LN
    - UI → E
    - UI → PP
    - UI → PC
    - UI → PR
    - UI → C
    - E → UI
    - E → P
    - E → A
    - E → FN
    - E → LN
    - E → PP
    - E → PC
    - E → PR
    - E → C
    - PC → PR
    - PC → C
    - A, PR, C → PC
2. Minimize LHS of each FD:
    - Nothing to reduce
3. Delete Redundant FD:

- UI → E
- E → UI
- E → P
- E → A
- E → FN
- E → LN
- E → PP
- E → PC
- PC → PR
- PC → C
- A, PR, C → PC

4. Minimal Cover
   - UI → E
   - E → UI
   - E → P
   - E → A
   - E → FN
   - E → LN
   - E → PP
   - E → PC
   - PC → PR
   - PC → C
   - A, PR, C → PC

Our FDs are not in 3NF, so we will decompose using the Lossless Join method:

1. A, PR, C → PC violates 3NF, decompose down to BCNF:
   - $R_1$(A, PR, C, PC)  $R_2$(UI, E, P, FN, LN, PP, A, PR, C)
2. PC → C violates BCNF, so decompose:
   - $R_3$(C, PC)  $R_4$(A, PR, PC)
3. PC → PR violates BCNF, so decompose:
   - $R_5$(A, PC)  $R_6$(PC, PR)
4. Missing A, PR, C → PC FD, so we add it back:
   - $R_7$(A, PR, C, PC)
5. Left with:
   - $R_2$(UI, E, P, FN, LN, PP, A, PR, C)
   - $R_3$(C, PC)
   - $R_5$(A, PC)
   - $R_6$(PC, PR)
   - $R_7$(A, PR, C, PC)
6. Optimize decomposition by removing $R_3$, $R_5$, and $R_6$ because they are subsets of $R_7$:
   - $R_2$(UI, E, P, FN, LN, PP, A, PR, C)
   - $R_7$(A, PR, C, PC)
7. Final set of relations are:
   - $R_2$(UI, E, P, FN, LN, PP, A, PR, C)

- $R_7$(A, PR, C, PC)

SCHEMAS:

UserInfo(

| | | |
|---|---|---|
| userID: integer | PK, | |
| email: char | UNIQUE NOT NULL, | |
| password: char | NOT NULL, | |
| firstName: char | NOT NULL, | |
| lastName: char | NOT NULL, | |
| profilePicture: char, | | |
| **address**: char | NOT NULL, | |
| **province**: char | NOT NULL, | |
| **city**: char | NOT NULL, | |
| CK(email), | | |
| FK(address, province, city) REF UserLocation | | |

)

UserLocation(

| | | |
|---|---|---|
| address: char | PK, | |
| province: char | PK, | |
| city: char | PK, | |
| postalCode: char | NOT NULL, | |
| PK(address, province, city) | | |

)

---

**COURT**

Give symbolic labels for each attribute to make normalization more readable:

- courtNumber → CN
- type → T
- address → A
- postalCode → PC
- surfaceMaterial → SM

Now we have the relation: Court(CN, T, A, PC, SM)

FDs for this relation:

- A, PC, CN → T, SM
- SM → T

Closures:

- $(A, PC, CN)^+ = \{A, PC, CN, T, SM\}$

- $(SM)^+ = \{SM, T\}$

Determine Minimal Key:

| Left | Middle | Right |
|---|---|---|
| A, PC, CN | SM | T |

Minimal Key: (A, PC, CN)

Finding Minimal Covers:

1. Put FDs into standard form:
    - A, PC, CN → T
    - A, PC, CN → SM
    - SM → T
2. Minimize LHS of each FD:
    - Nothing to minimize
3. Delete Redundant FD:
    - A, PC, CN → SM
    - SM → T
4. Minimal Cover:
    - A, PC, CN → SM
    - SM → T

Our FDs are not in 3NF, so we will decompose using the Synthesis method:

1. Create relations for each FD:
    - $R_1$(A, PC, CN, SM)
    - $R_2$(SM, T)
2. Minimal key not missing and no optimizations required, thus we are left with:
    - $R_1$(A, PC, CN, SM)
    - $R_2$(SM, T)

SCHEMAS:

Court(
courtNumber: integer PK,
**address**: char PK,
**postalCode**: char PK,
**surfaceMaterial**: char NOT NULL,
PK(courtNumber, address, postalCode),
FK(address, postalCode) REF Location(address, postalCode),
FK(surfaceMaterial) REF CourtMaterial,
)

CourtMaterial(

surfaceMaterial: char PK,
type: char NOT NULL,
)

## SQL DDL STATEMENTS

```
CREATE TABLE Location(
        address         VARCHAR(30),
        postalCode      CHAR(6),
        PRIMARY KEY(address, postalCode),
        FOREIGN KEY(postalCode) REFERENCES CityLocation
);

CREATE TABLE CityLocation(
        postalCode      CHAR(6)   PRIMARY KEY,
        city            VARCHAR(45)   NOT NULL
);

CREATE TABLE ProvinceLocation(
        postalCode      CHAR(6)   PRIMARY KEY,
        province        VARCHAR(20)   NOT NULL,
        FOREIGN KEY(postalCode) REFERENCES CityLocation
);

CREATE TABLE Court(
        courtNumber             INTEGER,
        surfaceMaterial         VARCHAR(20)   NOT NULL,
        address                 VARCHAR(30),
        postalCode              CHAR(6),
        PRIMARY KEY(courtNumber, address, postalCode),
        FOREIGN KEY(address, postalCode) REFERENCES Location(address, postalCode)
ON DELETE CASCADE,
        FOREIGN KEY(surfaceMaterial) REFERENCES CourtMaterial
);

CREATE TABLE CourtMaterial(
        surfaceMaterial         VARCHAR(20)   PRIMARY KEY,
        type                    VARCHAR(7)   NOT NULL
);

CREATE TABLE Singles(
        gameID              INTEGER   PRIMARY KEY,
        reservationID       INTEGER   UNIQUE   NOT NULL,
        gameInviteID        INTEGER   UNIQUE   NOT NULL,
        currentlyEnrolled   INTEGER   NOT NULL,
        capacity            INTEGER   NOT NULL,
        isActive            NUMBER(1)    NOT NULL,
        FOREIGN KEY(reservationID) REFERENCES Reservation ON DELETE CASCADE,
```

```
	FOREIGN KEY(gameInviteID) REFERENCES GameInvite ON DELETE CASCADE,
	FOREIGN KEY(currentlyEnrolled, capacity) REFERENCES
SinglesStatus(currentlyEnrolled, capacity)
);

CREATE TABLE SinglesStatus(
	currentlyEnrolled	INTEGER,
	capacity		INTEGER,
	isFull			NUMBER(1)   NOT NULL,
	PRIMARY KEY(currentlyEnrolled, capacity)
);

CREATE TABLE Doubles(
	gameID		INTEGER   PRIMARY KEY,
	reservationID	INTEGER   UNIQUE   NOT NULL,
	gameInviteID	INTEGER   UNIQUE   NOT NULL,
	currentlyEnrolled   INTEGER   NOT NULL,
	capacity	INTEGER   NOT NULL,
	isActive	NUMBER(1)    NOT NULL,
	FOREIGN KEY(reservationID) REFERENCES Reservation ON DELETE CASCADE,
	FOREIGN KEY(gameInviteID) REFERENCES GameInvite ON DELETE CASCADE,
	FOREIGN KEY(currentlyEnrolled, capacity) REFERENCES
DoublesStatus(currentlyEnrolled, capacity)
);

CREATE TABLE DoublesStatus(
	currentlyEnrolled	INTEGER,
	capacity		INTEGER,
	isFull			NUMBER(1)   NOT NULL,
	PRIMARY KEY(currentlyEnrolled, capacity)
);

CREATE TABLE Picture(
	pictureSrc	VARCHAR(4000)   PRIMARY KEY,
	altDescription  VARCHAR(200)
);

CREATE TABLE GameInvite(
	inviteID	INTEGER   PRIMARY KEY,
	status		NUMBER(1)   NOT NULL,
	description	VARCHAR(1000),
	thumbnail	VARCHAR(4000),
	creator		INTEGER   NOT NULL,
	sponsor		INTEGER,
```

```
        FOREIGN KEY(thumbnail) REFERENCES Picture ON DELETE SET NULL,
        FOREIGN KEY(creator) REFERENCES UserInfo ON DELETE CASCADE,
        FOREIGN KEY(sponsor) REFERENCES Club ON DELETE SET NULL
);

CREATE TABLE Club(
        clubID          INTEGER   PRIMARY KEY,
        name            VARCHAR(50)   NOT NULL
);

CREATE TABLE UserInfo(
        userID          INTEGER   PRIMARY KEY,
        email           VARCHAR(50)   UNIQUE   NOT NULL,
        password        VARCHAR(50)   NOT NULL,
        firstName       VARCHAR(30)   NOT NULL,
        lastName        VARCHAR(30)   NOT NULL,
        profilePicture  VARCHAR(4000),
        address         VARCHAR(30)   NOT NULL,
        province        VARCHAR(20)   NOT NULL,
        city            VARCHAR(45)   NOT NULL,
        FOREIGN KEY(address, province, city) REFERENCES UserLocation(address, province,
city)
);

CREATE TABLE UserLocation(
        address         VARCHAR(30),
        province        VARCHAR(20),
        city            VARCHAR(45),
        postalCode      CHAR(6)   NOT NULL,
        PRIMARY KEY(address, province, city)
);

CREATE TABLE Reservation(
        reservationID   INTEGER   PRIMARY KEY,
        bookingTime     DATE,
        courtNumber     INTEGER   NOT NULL,
        address         VARCHAR(30)   NOT NULL,
        postalCode      CHAR(6)   NOT NULL,
        UNIQUE(bookingTime, courtNumber, address, postalCode),
        FOREIGN KEY(courtNumber, address, postalCode) REFERENCES Court(courtNumber,
address, postalCode) ON DELETE CASCADE
);

CREATE TABLE Comments(
```

```
        commentID           INTEGER   PRIMARY KEY,
        content             VARCHAR(500)   NOT NULL,
        commentedBy         INTEGER   NOT NULL,
        FOREIGN KEY(commentedBy) REFERENCES UserInfo ON DELETE CASCADE
);

CREATE TABLE replyTo(
        parentID        INTEGER,
        replyID         INTEGER,
        PRIMARY KEY (parentID, replyID),
        FOREIGN KEY (parentID) REFERENCES Comments ON DELETE CASCADE,
        FOREIGN KEY (replyID) REFERENCES Comments ON DELETE CASCADE
);

CREATE TABLE commentedOn(
        commentID    INTEGER,
        inviteID        INTEGER,
        PRIMARY KEY (commentID, InviteID),
        FOREIGN KEY (commentID) REFERENCES Comments ON DELETE CASCADE,
        FOREIGN KEY (inviteID) REFERENCES GameInvite ON DELETE CASCADE
);

CREATE TABLE apartOf(
        userID          INTEGER,
        clubID          INTEGER,
        PRIMARY KEY (userID, clubID),
        FOREIGN KEY (userID) REFERENCES UserInfo ON DELETE CASCADE,
        FOREIGN KEY (clubID) REFERENCES Club ON DELETE CASCADE
);

CREATE TABLE registers(
        userID          INTEGER,
        inviteID        INTEGER,
        PRIMARY KEY (userID, inviteID),
        FOREIGN KEY (userID) REFERENCES UserInfo ON DELETE CASCADE,
        FOREIGN KEY (inviteID) REFERENCES GameInvite ON DELETE CASCADE
);
```

## INSERT STATEMENTS

```
INSERT INTO Location(address, postalCode) VALUES ('123 Sports Ave', 'V23AB1');
INSERT INTO Location(address, postalCode) VALUES ('509 Pickle St', 'V21PL7');
INSERT INTO Location(address, postalCode) VALUES ('123 Ball Rd', 'VL920N');
INSERT INTO Location(address, postalCode) VALUES ('540 Court Rd', 'V2BLO1');
INSERT INTO Location(address, postalCode) VALUES ('752 Racket St', 'V1PL77');

INSERT INTO CityLocation(postalCode, City) VALUES ('V23AB1', 'Vancouver');
INSERT INTO CityLocation(postalCode, City) VALUES ('V21PL7', 'Vancouver');
INSERT INTO CityLocation(postalCode, City) VALUES ('VL920N', 'Vancouver');
INSERT INTO CityLocation(postalCode, City) VALUES ('V2BLO1', 'Vancouver');
INSERT INTO CityLocation(postalCode, City) VALUES ('V1PL77', 'Vancouver');

INSERT INTO ProvinceLocation(postalCode, province) VALUES ('V23AB1', 'BC');
INSERT INTO ProvinceLocation(postalCode, province) VALUES ('V21PL7', 'BC');
INSERT INTO ProvinceLocation(postalCode, province) VALUES ('VL920N', 'BC');
INSERT INTO ProvinceLocation(postalCode, province) VALUES ('V2BLO1', 'BC');
INSERT INTO ProvinceLocation(postalCode, province) VALUES ('V1PL77', 'BC');

INSERT INTO Court(courtNumber, surfaceMaterial, address, postalCode) VALUES (1,
'Hardwood',  '123 Sports Ave', 'V23AB1');
INSERT INTO Court(courtNumber, surfaceMaterial, address, postalCode) VALUES (1, 'Rubber',
'509 Pickle St', 'V21PL7');
INSERT INTO Court(courtNumber, surfaceMaterial, address, postalCode) VALUES (1, 'Plastic
Tiles', '123 Ball Rd', 'VL920N');
INSERT INTO Court(courtNumber, surfaceMaterial, address, postalCode) VALUES (1, 'Cement',
'540 Court Rd', 'V2BLO1');
INSERT INTO Court(courtNumber, surfaceMaterial, address, postalCode) VALUES (1, 'Asphalt',
'752 Racket St', 'V1PL77');

INSERT INTO CourtMaterial(surfaceMaterial, type) VALUES ('Hardwood', 'Indoor');
INSERT INTO CourtMaterial(surfaceMaterial, type) VALUES ('Rubber', 'Indoor');
INSERT INTO CourtMaterial(surfaceMaterial, type) VALUES ('Plastic Tiles', 'Indoor');
INSERT INTO CourtMaterial(surfaceMaterial, type) VALUES ('Cement', 'Outdoor');
INSERT INTO CourtMaterial(surfaceMaterial, type) VALUES ('Asphalt', 'Outdoor');

INSERT INTO Singles(gameId, reservationID, gameInviteID, currentlyEnrolled, capacity,
isActive) VALUES (1, 1, 1, 0, 2, 0);
INSERT INTO Singles(gameId, reservationID, gameInviteID, currentlyEnrolled, capacity,
isActive) VALUES (2, 2, 2, 0, 6, 0);
INSERT INTO Singles(gameId, reservationID, gameInviteID, currentlyEnrolled, capacity,
isActive) VALUES (3, 3, 3, 0, 3, 0);
INSERT INTO Singles(gameId, reservationID, gameInviteID, currentlyEnrolled, capacity,
isActive) VALUES (4, 4, 4, 0, 4, 0);
```

```
INSERT INTO Singles(gameId, reservationID, gameInviteID, currentlyEnrolled, capacity,
isActive) VALUES (5, 5, 5, 0, 5, 0);

INSERT INTO SinglesStatus(currentlyEnrolled, capacity, isFull) VALUES (0, 2, 0);
INSERT INTO SinglesStatus(currentlyEnrolled, capacity, isFull) VALUES (0, 3, 0);
INSERT INTO SinglesStatus(currentlyEnrolled, capacity, isFull) VALUES (0, 4, 0);
INSERT INTO SinglesStatus(currentlyEnrolled, capacity, isFull) VALUES (0, 5, 0);
INSERT INTO SinglesStatus(currentlyEnrolled, capacity, isFull) VALUES (0, 6, 0);

INSERT INTO Doubles(gameId, reservationID, gameInviteID, currentlyEnrolled, capacity,
isActive) VALUES (1, 6, 6, 0, 2, 0);
INSERT INTO Doubles(gameId, reservationID, gameInviteID, currentlyEnrolled, capacity,
isActive) VALUES (2, 7, 7, 0, 3, 0);
INSERT INTO Doubles(gameId, reservationID, gameInviteID, currentlyEnrolled, capacity,
isActive) VALUES (3, 8, 8, 0, 4, 0);
INSERT INTO Doubles(gameId, reservationID, gameInviteID, currentlyEnrolled, capacity,
isActive) VALUES (4, 9, 9, 0, 5, 0);
INSERT INTO Doubles(gameId, reservationID, gameInviteID, currentlyEnrolled, capacity,
isActive) VALUES (5, 10, 10, 0, 6, 0);

INSERT INTO DoublesStatus(currentlyEnrolled, capacity, isFull) VALUES (0, 2, 0);
INSERT INTO DoublesStatus(currentlyEnrolled, capacity, isFull) VALUES (0, 3, 0);
INSERT INTO DoublesStatus(currentlyEnrolled, capacity, isFull) VALUES (0, 4, 0);
INSERT INTO DoublesStatus(currentlyEnrolled, capacity, isFull) VALUES (0, 5, 0);
INSERT INTO DoublesStatus(currentlyEnrolled, capacity, isFull) VALUES (0, 6, 0);

INSERT INTO Picture (pictureSrc, altDescription) VALUES
('https://m.media-amazon.com/images/I/71PvykgfktL._AC_SX679_.jpg', 'Raquets');
INSERT INTO Picture (pictureSrc, altDescription) VALUES
('https://pickleballsuperstore.com/cdn/shop/articles/pickleball_court_dimensions_top_1000x.jpg?
v=1642839067', 'Courts');
INSERT INTO Picture (pictureSrc, altDescription) VALUES
('https://m.media-amazon.com/images/I/61Ai-DOZmEL.__AC_SX300_SY300_QL70_ML2_.jpg',
'Pickleballs');
INSERT INTO Picture (pictureSrc, altDescription) VALUES
('https://www.burnaby.ca/sites/default/files/styles/rad_classic_1200w/public/acquiadam/2021-08/
Keswick%20Park%20Courts%201280%20x%20720.JPG?h=c673cd1c', 'Outdoor Courts');
INSERT INTO Picture (pictureSrc, altDescription) VALUES
('https://m.media-amazon.com/images/I/81Xyf1liAgL.__AC_SX300_SY300_QL70_ML2_.jpg',
NULL);

INSERT INTO GameInvite (inviteID, status, description, thumbnail, creator, sponsor) VALUES
(1, 1, NULL,
```

'https://m.media-amazon.com/images/I/81Xyf1liAgL.__AC_SX300_SY300_QL70_ML2_.jpg', 1, NULL);
INSERT INTO GameInvite (inviteID, status, description, thumbnail, creator, sponsor) VALUES (2, 1, NULL, 'https://m.media-amazon.com/images/I/71PvykgfktL._AC_SX679_.jpg', 1, 3);
INSERT INTO GameInvite (inviteID, status, description, thumbnail, creator, sponsor) VALUES (3, 0, 'New Pickleball Game', NULL, 2, NULL);
INSERT INTO GameInvite (inviteID, status, description, thumbnail, creator, sponsor) VALUES (4, 1, NULL, NULL, 3, 5);
INSERT INTO GameInvite (inviteID, status, description, thumbnail, creator, sponsor) VALUES (5, 0, 'Hello', NULL, 5, 2);
INSERT INTO GameInvite (inviteID, status, description, thumbnail, creator, sponsor) VALUES (6, 1, NULL,
'https://m.media-amazon.com/images/I/81Xyf1liAgL.__AC_SX300_SY300_QL70_ML2_.jpg', 1, NULL);
INSERT INTO GameInvite (inviteID, status, description, thumbnail, creator, sponsor) VALUES (7, 1, NULL, 'https://m.media-amazon.com/images/I/71PvykgfktL._AC_SX679_.jpg', 1, 3);
INSERT INTO GameInvite (inviteID, status, description, thumbnail, creator, sponsor) VALUES (8, 0, 'New Pickleball Game', NULL, 2, NULL);
INSERT INTO GameInvite (inviteID, status, description, thumbnail, creator, sponsor) VALUES (9, 1, NULL, NULL, 3, 5);
INSERT INTO GameInvite (inviteID, status, description, thumbnail, creator, sponsor) VALUES (10, 0, 'Hello', NULL, 5, 2);

INSERT INTO Club (clubID, name) VALUES (1, 'Pickleballers');
INSERT INTO Club (clubID, name) VALUES (2, 'The Pickles');
INSERT INTO Club (clubID, name) VALUES (3, 'Racket Lovers');
INSERT INTO Club (clubID, name) VALUES (4, 'Ball');
INSERT INTO Club (clubID, name) VALUES (5, 'Orange');

INSERT INTO UserInfo (userID, email, password, firstName, lastName, profilePicture, address, province, city) VALUES (1, 'test1@gmail.com', 'apple', 'John', 'Smith', NULL, '111 Apple St', 'BC', 'Vancouver');
INSERT INTO UserInfo (userID, email, password, firstName, lastName, profilePicture, address, province, city) VALUES (2, 'test2@gmail.com', 'lime', 'Joe', 'Smith', NULL, '111 Lime St', 'BC', 'Vancouver');
INSERT INTO UserInfo (userID, email, password, firstName, lastName, profilePicture, address, province, city) VALUES (3, 'test3@gmail.com', 'banana', 'Jane', 'Doe', NULL, '111 Banana St', 'BC', 'Vancouver');
INSERT INTO UserInfo (userID, email, password, firstName, lastName, profilePicture, address, province, city) VALUES (4, 'test4@gmail.com', 'blueberry', 'Lance', 'Jones', NULL, '111 Blueberry St', 'BC', 'Vancouver');
INSERT INTO UserInfo (userID, email, password, firstName, lastName, profilePicture, address, province, city) VALUES (5, 'test5@gmail.com', 'grape', 'Aria', 'Bell', NULL, '111 Grape St', 'BC', 'Vancouver');

```
INSERT INTO UserLocation (address, province, city, postalCode) VALUES ('111 Apple St', 'BC', 'Vancouver', 'V12NOP');
INSERT INTO UserLocation (address, province, city, postalCode) VALUES ('111 Lime St', 'BC', 'Vancouver', 'V15TBH');
INSERT INTO UserLocation (address, province, city, postalCode) VALUES ('111 Banana St', 'BC', 'Vancouver', 'V63PLE');
INSERT INTO UserLocation (address, province, city, postalCode) VALUES ('111 Blueberry St', 'BC', 'Vancouver', 'V90MN1');
INSERT INTO UserLocation (address, province, city, postalCode) VALUES ('111 Grape St', 'BC', 'Vancouver', 'VT1GH4');

INSERT INTO Reservation (reservationID, bookingTime, courtNumber, address, postalCode) VALUES (1, TO_DATE('2024/11/01 15:00', 'yyyy/mm/dd hh24:mi'), 1, '123 Sports Ave', 'V23AB1');
INSERT INTO Reservation (reservationID, bookingTime, courtNumber, address, postalCode) VALUES (2, TO_DATE('2024/11/02 15:00', 'yyyy/mm/dd hh24:mi'), 1, '123 Sports Ave', 'V23AB1');
INSERT INTO Reservation (reservationID, bookingTime, courtNumber, address, postalCode) VALUES (3, TO_DATE('2024/11/01 20:00', 'yyyy/mm/dd hh24:mi'), 1, '509 Pickle St', 'V21PL7');
INSERT INTO Reservation (reservationID, bookingTime, courtNumber, address, postalCode) VALUES (4, TO_DATE('2024/11/02 15:00', 'yyyy/mm/dd hh24:mi'), 1, '509 Pickle St', 'V21PL7');
INSERT INTO Reservation (reservationID, bookingTime, courtNumber, address, postalCode) VALUES (5, TO_DATE('2024/11/01 10:00', 'yyyy/mm/dd hh24:mi'), 1, '123 Ball Rd', 'VL920N');
INSERT INTO Reservation (reservationID, bookingTime, courtNumber, address, postalCode) VALUES (6, TO_DATE('2024/11/01 15:00', 'yyyy/mm/dd hh24:mi'), 1, '540 Court Rd', 'V2BLO1');
INSERT INTO Reservation (reservationID, bookingTime, courtNumber, address, postalCode) VALUES (7, TO_DATE('2024/11/02 10:00', 'yyyy/mm/dd hh24:mi'), 1, '540 Court Rd', 'V2BLO1');
INSERT INTO Reservation (reservationID, bookingTime, courtNumber, address, postalCode) VALUES (8, TO_DATE('2024/11/01 20:00', 'yyyy/mm/dd hh24:mi'), 1, '752 Racket St', 'V1PL77');
INSERT INTO Reservation (reservationID, bookingTime, courtNumber, address, postalCode) VALUES (9, TO_DATE('2024/11/02 15:00', 'yyyy/mm/dd hh24:mi'), 1, '752 Racket St', 'V1PL77');
INSERT INTO Reservation (reservationID, bookingTime, courtNumber, address, postalCode) VALUES (10, TO_DATE('2024/11/02 10:00', 'yyyy/mm/dd hh24:mi'), 1, '123 Ball Rd', 'VL920N');

INSERT INTO Comments (commentID, content, commentedBy) VALUES (1, 'Hello', 1);
INSERT INTO Comments (commentID, content, commentedBy) VALUES (2, 'Goodbye', 1);
INSERT INTO Comments (commentID, content, commentedBy) VALUES (3, 'Yes', 3);
INSERT INTO Comments (commentID, content, commentedBy) VALUES (4, 'No', 4);
INSERT INTO Comments (commentID, content, commentedBy) VALUES (5, 'Apple', 2);
INSERT INTO Comments (commentID, content, commentedBy) VALUES (6, 'Banana', 5);

INSERT INTO replyTo(parentID, replyID) VALUES (1, 2);
INSERT INTO replyTo(parentID, replyID) VALUES (1, 3);
```

```
INSERT INTO replyTo(parentID, replyID) VALUES (1, 4);
INSERT INTO replyTo(parentID, replyID) VALUES (1, 5);
INSERT INTO replyTo(parentID, replyID) VALUES (1, 6);

INSERT INTO commentedOn(commentID, inviteID) VALUES (1, 5);
INSERT INTO commentedOn(commentID, inviteID) VALUES (2, 3);
INSERT INTO commentedOn(commentID, inviteID) VALUES (3, 2);
INSERT INTO commentedOn(commentID, inviteID) VALUES (4, 2);
INSERT INTO commentedOn(commentID, inviteID) VALUES (5, 1);

INSERT INTO apartOf(userID, clubID) VALUES (1, 1);
INSERT INTO apartOf(userID, clubID) VALUES (1, 3);
INSERT INTO apartOf(userID, clubID) VALUES (3, 2);
INSERT INTO apartOf(userID, clubID) VALUES (4, 2);
INSERT INTO apartOf(userID, clubID) VALUES (5, 4);

INSERT INTO registers(userID, inviteID) VALUES (1, 5);
INSERT INTO registers(userID, inviteID) VALUES (2, 4);
INSERT INTO registers(userID, inviteID) VALUES (3, 3);
INSERT INTO registers(userID, inviteID) VALUES (4, 3);
INSERT INTO registers(userID, inviteID) VALUES (5, 2);
```