Part 2: Logical Database Design: Mapping ER Design to the Relational Model

```
CREATE TABLE Channels(
        channel_ID INTEGER,
        channel_name CHAR(20),
        PRIMARY KEY (channel_ID));
CREATE TABLE Positions(
        position_ID INTEGER,
        position_name CHAR(20),
        PRIMARY KEY (position_ID));
CREATE TABLE Stadium(
        stadium_ID INTEGER,
        stadium_name CHAR(20),
        stadium_country CHAR(20),
        PRIMARY KEY (stadium_ID));
CREATE TABLE Users (
        username CHAR(20),
        passwords CHAR(20),
        name_ CHAR(20),
        surname CHAR(20),
        PRIMARY KEY (username));
CREATE TABLE Player(
        username CHAR(20),
        date_of_birth DATE,
        height INTEGER,
        weight INTEGER,
        PRIMARY KEY (username),
        FOREIGN KEY (username)
                REFERENCES Users(username)
                        ON DELETE CASCADE);
```

```
CREATE TABLE Jury(
        username CHAR(20),
        nationality CHAR(20),
        PRIMARY KEY (username),
        FOREIGN KEY (username)
                REFERENCES Users(username)
                        ON DELETE CASCADE);
CREATE TABLE Coach(
        username CHAR(20),
        nationality CHAR(20),
        PRIMARY KEY (username),
        FOREIGN KEY (username)
                REFERENCES Users(username)
                        ON DELETE CASCADE);
CREATE TABLE Team(
        team_ID INTEGER,
        team_name CHAR(20),
        channel_ID INTEGER NOT NULL,
        PRIMARY KEY (team_ID),
        FOREIGN KEY (channel_ID)
                REFERENCES Channels(channel_ID));
```

```
# when a team changes a coach this has to be deleted before new insertion
CREATE TABLE Directs(
        username CHAR(20),
        team_ID INTEGER,
        contract_start DATE NOT NULL,
  contract_finish DATE NOT NULL,
        UNIQUE (username),
  PRIMARY KEY (team_ID),
        FOREIGN KEY (username)
                REFERENCES Coach(username)
                        ON DELETE CASCADE,
        FOREIGN KEY (team_ID)
                REFERENCES Team(team_ID)
                        ON DELETE CASCADE,
        CHECK (contract_start <= contract_finish));

CREATE TABLE Playsposition(
        position_ID INTEGER,
        username CHAR(20),
        PRIMARY KEY (position_ID, username),
        FOREIGN KEY (position_ID)
                REFERENCES Positions(position_ID)
                        ON DELETE CASCADE,
        FOREIGN KEY (username)
                REFERENCES Player(username)
                        ON DELETE CASCADE);
```

```
CREATE TABLE Playsinteam(
        username CHAR(20),
        team_ID INTEGER,
        PRIMARY KEY (team_ID, username),
        FOREIGN KEY (team_ID)
                REFERENCES Team(team_ID)
                        ON DELETE CASCADE,
        FOREIGN KEY (username)
                REFERENCES Player(username)
                        ON DELETE CASCADE);
CREATE TABLE Matchs(
        session_ID INTEGER,
        match_date DATE NOT NULL,
        time_slot INTEGER NOT NULL,
        stadium_ID INTEGER NOT NULL,
        username CHAR(20),
        rating INTEGER,
        PRIMARY KEY (session_ID),
        FOREIGN KEY (stadium_ID)
                REFERENCES Stadium(stadium_ID),
        FOREIGN KEY (username)
                REFERENCES Jury(username),
        CHECK (time_slot > 0 and time_slot < 4));
```

```
CREATE TABLE Plays(
        session_ID INTEGER,
        username CHAR(20),
        team_ID INTEGER,
        position_ID INTEGER,
        PRIMARY KEY (session_ID, team_ID, username),
        FOREIGN KEY (team_ID, username)
                REFERENCES Playsinteam(team_ID, username),
        FOREIGN KEY (position_ID)
                REFERENCES Positions(position_ID),
        FOREIGN KEY (session_ID)
                REFERENCES Matchs(session_ID)
#CHECK ((SELECT COUNT(*) FROM Playsposition P
        #WHERE P.username = username and P.position_ID = position_ID) > 0)
);
```

Discussion of Part 2

In the coding part, some constraints could not be captured. These constraints are:

1. Total participation of *"Player"* entity in the *"plays_position"* relation with *"Position"*. This constraint can be shown in the ER Design but cannot be enforced in the CREATE TABLE command (it is also the case in the lecture slides).
2. Total participation of *"Player"* entity in the *"plays_team"* relation with *"Team"*. This constraint can be shown in the ER Design but cannot be enforced in the CREATE TABLE command (it is also the case in the lecture slides).
3. Two different matches cannot happen at the same time and place. This requires assertion to be enforced and MySQL does not support assertion so it cannot be enforced.
4. In reality, a team can agree with a coach for a while and after that, the team can agree with another coach (when the previous coach's contract expires). But in the code, this cannot be shown. Each team has a single and unique coach. If the teams were allowed to have more than one coach or a single coach were allowed to coach different teams, we could not impose a mechanism to prevent overlapping contracts, and this would result in a violation of the principle stating that each team is directed by a unique coach between contract start and contract end date. This requires assertion to be enforced and MySQL does not support assertion so it cannot be enforced.
5. The *"plays"* relation between aggregation of "*plays_team"* relation between *"Player"* entity and "*Team"* entity does not disallow a player to be in two matches that happen at the same time and day. It does not prevent the possibility of the players having time conflict between their matches. This requires assertion to be enforced and MySQL does not support assertion so it cannot be enforced.
6. The *"plays"* relation between aggregation of "*plays_team"* relation between *"Player"* entity and "*Team"* entity allows players from different teams to play in the same match but this must not happen. This requires assertion to be enforced and MySQL does not support assertion so it cannot be enforced.
7. The *"plays"* relation between aggregation of "*plays_team"* relation between *"Player"* entity and "*Team"* entity does not check if a player can be at the given position (denoted using *"position_ID"* attribute of *"plays"* relation) at a match. The position ID list of the player must include *"position_ID"* attribute of *"plays"* relation. We created a check similar to lecture slides to handle this, but it did not work, we think it might be because of MySQL syntax, so we left it as a comment.
8. Juries cannot change their ratings, but we are not able to implement this.