-- Table to store members general information

CREATE TABLE MemberGen (

idnumber INT PRIMARY KEY,

name VARCHAR(100),

surname VARCHAR(100),

iban VARCHAR(34),

email VARCHAR(100),

FOREIGN KEY (idnumber) REFERENCES MemberLoc(idnumber)

);

-- Table to store member's location information

CREATE TABLE MemberLoc (

idnumber INT PRIMARY KEY,

city VARCHAR(100),

postalcode VARCHAR(10),

street VARCHAR(100),

housenumber VARCHAR(10),

FOREIGN KEY (idnumber) REFERENCES MemberGen(idnumber)

);

-- Table to store telephone information of members

CREATE TABLE TelPer (

member INT,

telephoneNumber VARCHAR(15),

PRIMARY KEY (member, telephoneNumber),

FOREIGN KEY (member) REFERENCES MemberGen(idnumber),

FOREIGN KEY (telephoneNumber) REFERENCES TelephoneNumber(number)

);

-- Table to store valid telephone numbers

CREATE TABLE TelephoneNumber (

number VARCHAR(15) PRIMARY KEY

);

-- Table to store active members

CREATE TABLE ActiveMember (

member INT PRIMARY KEY,

FOREIGN KEY (member) REFERENCES MemberGen(idnumber)

);

-- Table to store managerial members

CREATE TABLE ManagerialMember (

member INT PRIMARY KEY,

FOREIGN KEY (member) REFERENCES MemberGen(idnumber)

);

-- Table to store observers

CREATE TABLE Observer (

activeMember INT PRIMARY KEY,

FOREIGN KEY (activeMember) REFERENCES ActiveMember(member)

);

-- Table to store referees

CREATE TABLE Referee (

activeMember INT PRIMARY KEY,

FOREIGN KEY (activeMember) REFERENCES ActiveMember(member)

);

-- Table to store unavailability of active members

CREATE TABLE Unavailability (

code INT PRIMARY KEY,

startdate DATE,

enddate DATE,

activeMember INT,

FOREIGN KEY (activeMember) REFERENCES ActiveMember(member)

);

-- Table to store leaves

CREATE TABLE Leave (

unavailability INT PRIMARY KEY,

description TEXT,

FOREIGN KEY (unavailability) REFERENCES Unavailability(code)

);

-- Table to link leaves with active members

CREATE TABLE Leaves (

leave INT,

activeMember INT,

PRIMARY KEY (leave, activeMember),

FOREIGN KEY (leave) REFERENCES Leave(unavailability),

FOREIGN KEY (activeMember) REFERENCES ActiveMember(member)

);

-- Table to store medical certificates for referees

CREATE TABLE MedicalCertificate (

code INT PRIMARY KEY,

startdate DATE,

enddate DATE,

referee INT,

FOREIGN KEY (referee) REFERENCES Referee(activeMember)

);

-- Table to store observer assignations

CREATE TABLE ObserverAssignation (

match INT,

pay DECIMAL(10, 2),

state VARCHAR(50),

observer INT,

PRIMARY KEY (match, observer),

FOREIGN KEY (observer) REFERENCES Observer(activeMember),

FOREIGN KEY (match) REFERENCES Match(code)

);

-- Table to store referee assignations

CREATE TABLE RefereeAssignation (

match INT,

pay DECIMAL(10, 2),

state VARCHAR(50),

referee INT,

PRIMARY KEY (match, referee),

FOREIGN KEY (referee) REFERENCES Referee(activeMember),

FOREIGN KEY (match) REFERENCES Match(code)

);

-- Table to store managerial members managing matches

CREATE TABLE Manages (

match INT,

managerialMember INT,

PRIMARY KEY (match, managerialMember),

FOREIGN KEY (match) REFERENCES Match(code),

FOREIGN KEY (managerialMember) REFERENCES ManagerialMember(member)

);

-- Table to store matches

CREATE TABLE Match (

code INT PRIMARY KEY,

location VARCHAR(100),

date DATE,

competitionName VARCHAR(100)

);

-- Table to store referee exemptions from teams/categories

CREATE TABLE IsExempted (

referee INT,

team VARCHAR(100),

category VARCHAR(50),

PRIMARY KEY (referee, team, category),

FOREIGN KEY (referee) REFERENCES Referee(activeMember),

FOREIGN KEY (team, category) REFERENCES Team(name, category)

);

-- Table to store team information

CREATE TABLE Team (

name VARCHAR(100),

category VARCHAR(50),

PRIMARY KEY (name, category)

);

-- Table to store match plays between teams

CREATE TABLE Plays (

match INT,

homeTeamName VARCHAR(100),

homeTeamCategory VARCHAR(50),

guestTeamName VARCHAR(100),

guestTeamCategory VARCHAR(50),

PRIMARY KEY (match),

FOREIGN KEY (homeTeamName, homeTeamCategory) REFERENCES Team(name, category),

FOREIGN KEY (guestTeamName, guestTeamCategory) REFERENCES Team(name, category)

);

-- External Constraints as Triggers:

-- 1) Referee cannot be assigned without a valid medical certificate on match date

CREATE OR REPLACE FUNCTION check\_referee\_medical\_certificate() RETURNS TRIGGER AS $$

BEGIN

-- Ensure the referee has a valid medical certificate during the match date

IF NOT EXISTS (

SELECT 1 FROM MedicalCertificate

WHERE referee = NEW.referee

AND (SELECT date FROM Match WHERE code = NEW.match) BETWEEN startdate AND enddate

) THEN

RAISE EXCEPTION '';

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trg\_check\_referee\_medical\_certificate

BEFORE INSERT OR UPDATE ON RefereeAssignation

FOR EACH ROW EXECUTE FUNCTION check\_referee\_medical\_certificate();

-- 2) Referee cannot be assigned if unavailable on match date

CREATE OR REPLACE FUNCTION check\_referee\_unavailability() RETURNS TRIGGER AS $$

BEGIN

-- Ensure the referee is not unavailable during the match date

IF EXISTS (

SELECT 1 FROM Unavailability

WHERE activeMember = NEW.referee

AND (SELECT date FROM Match WHERE code = NEW.match) BETWEEN startdate AND enddate

) THEN

RAISE EXCEPTION '';

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trg\_check\_referee\_unavailability

BEFORE INSERT OR UPDATE ON RefereeAssignation

FOR EACH ROW EXECUTE FUNCTION check\_referee\_unavailability();

-- 3) Referee cannot be assigned to a match involving a team they are exempted from officiating

CREATE OR REPLACE FUNCTION check\_referee\_exemption() RETURNS TRIGGER AS $$

BEGIN

-- Ensure the referee is not exempted from the match teams

IF EXISTS (

SELECT 1 FROM IsExempted

WHERE referee = NEW.referee

AND (team = (SELECT homeTeamName FROM Plays WHERE match = NEW.match)

OR team = (SELECT guestTeamName FROM Plays WHERE match = NEW.match))

) THEN

RAISE EXCEPTION '';

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trg\_check\_referee\_exemption

BEFORE INSERT OR UPDATE ON RefereeAssignation

FOR EACH ROW EXECUTE FUNCTION check\_referee\_exemption();