Filosi Davide (22760), Muhammad Kashif
Modelling and Database Project
September 2024
Football Referee Club

Conceptual design

Requirements specification

Introduction
We want to realize a database for a Football Referee Club, which is responsible for the refereeing of some football matches.

Description
For each member of the club, identified by the id number, we want to store the name, the surname, the email, the iban code, the phone number (at least one and maximum two) and the address, that is composed of the city, the postal code, the street and the house number.
A member can be an active member, a managerial member, both or neither of the two.
Then an active member can only be either a referee or an observer. An active member can be related to an unavailability, of which we want to represent the code, the start date and the end date. An active member can also be related to a leave that is just an unavailability but with a detailed description.
An observer is assigned to an observer assignation, of which we want to represent the state and the pay. A match can have at most one observer assignation.
Similarly to the observer, a referee is assigned to a referee assignation, of which we are interested in the state and the pay. Then, a referee can be exempted for a certain team and has at least one medical certificate, of which the following are of interest: code, start date and end date.
A managerial member is responsible for managing the matches. Of a match the following are of interest: code, location, date and the competition name. A match has one and only one referee assignation and is related to two teams: the home team and the guest team. A team is identified by both the name and the category.

An active member cannot be assigned to a match if on the date of the match he is unavailable or has not a valid medical certificate. He can not also be assigned to a match played by one or two teams for which he is exempted.

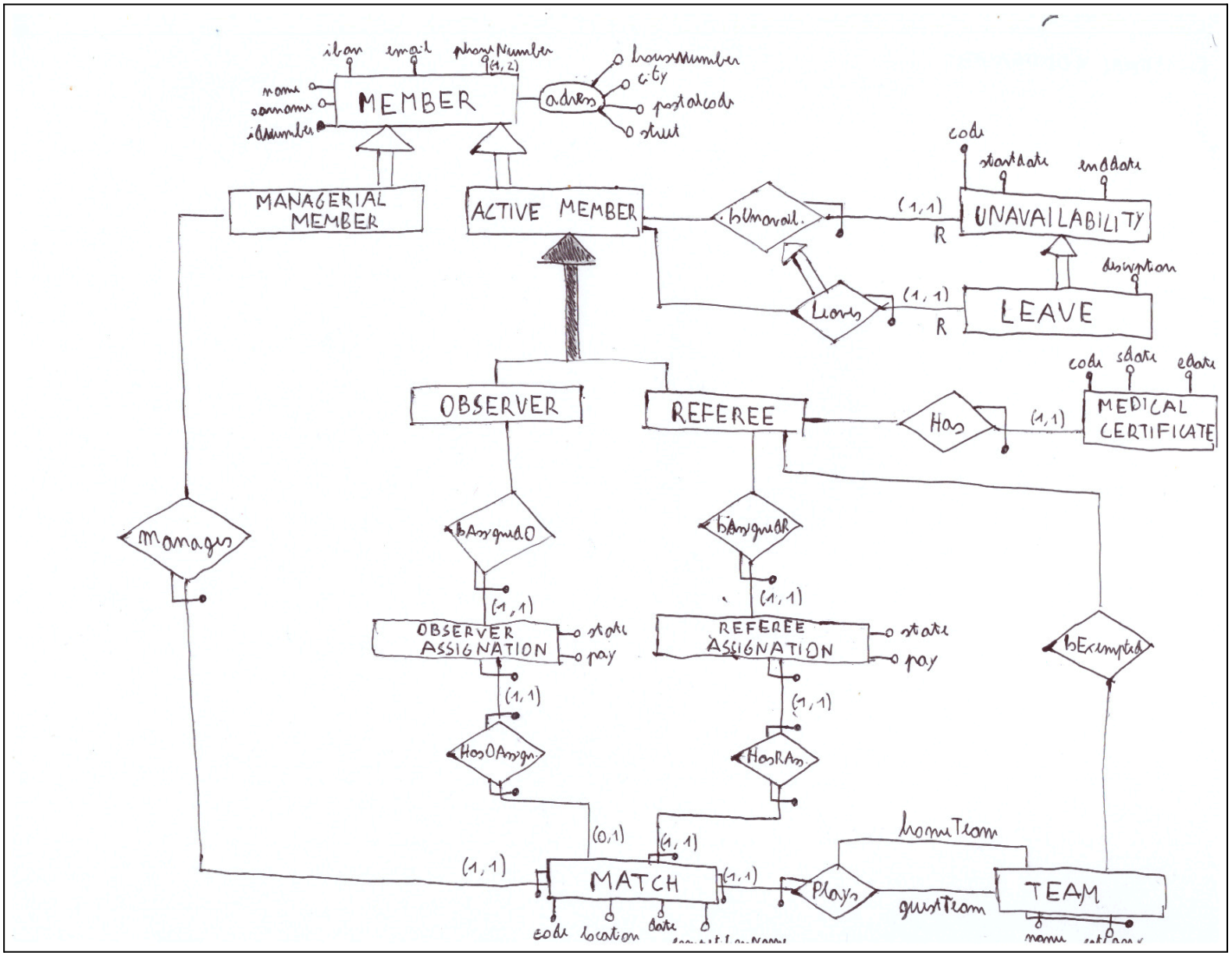
Scope
The main goal of the system is to store the data needed to assign matches to active members taking into account exemptions, unavailabilities and medical certificates of each member.

Glossary of terms

Term	Description	Synonyms	Connections
Member	Person who is part of the Referee club		Managerial member, active member
Managerial member	Member who assigns referees and observers to matches	Designator, appointer	Match
Active member	Member who is either a referee or an observer		Unavailability, Leave
Unavailability	A period during which an active member is unavailable for refereeing		Leave, Active member
Leave	Similar to Unavailability but with a precise description		Unavailability, Active member
Referee	Active member whose role is the referee		Active member, Medical certificate, Team, Referee assignation
Medical certificate	Document that ensures that the referee is adequately fit for refereeing		Referee
Referee assignation	It relates a referee with a match (the referee will be the referee in the game he is assigned to)		Referee, Match
Observer	Active member who goes to watch (observe) a referee during a match. He then gives the referee advices to improve and an evaluation	Judge/Coach	Active member, Observer assignation
Observer assignation	It relates an observer with a match		Observer, Match

Match	A football match		Managerial member, Observer assignation, Referee assignation, Team
Team	A football team that is of a certain society and category		Referee, Match

ER schema



(competitionname) (category)

External constraints

1	A referee can not be assigned to a match, if on the day of the match date the referee has no valid medical certificate.
2	A referee can not be assigned to a match, if on the day of the match date the referee is unavailable (is related to an unavailability).
3	A referee can not be assigned to a match that involves a team for which he asked to be exempted.

Table of volumes

Concept	Construct	Volume
Member	Entity	55
ManagerialMember	Entity	1
ActiveMember	Entity	50
Referee	Entity	40
Observer	Entity	10
Unavailability	Entity	100
Leave	Entity	2
MedicalCertificate	Entity	50
RefereeAssignation	Entity	200
ObserverAssignation	Entity	*50*
Match	Entity	200
Team	Entity	200
IsUnavailable	Relationship	100
Leaves	Relationship	2
Has	Relationship	50
IsExempted	Relationship	40
Manages	Relationship	200
IsAssignedReferee	Relationship	200
IsAssignedObsever	Relationship	50
HasRefereeAssignation	Relationship	200
HasObserverAssignation	Relationship	50
Plays	Relationship	200

Operations of interest are:

1. Retrieve suitable referees for a certain match, that are referees who
    - have a valid medical certificate in the match date
    - are available on the match date
    - are not exempted from one or both teams of the match
    - have not already an assignation for a match on the same date
2. Retrieve suitable observers for a certain match, that are observers who:
    - are available on the match date
    - have not already an assignation for a match on the same date
3. Assign a match to a referee.
4. Assign a match to an observer
5. Update the state of an assignation
6. Retrieve member's iban or residency location.

Table of operations

Op.	Type	Frequency
1	Interactive	40/week
2	Interactive	10/week
3	Interactive	40/week
4	Interactive	10/week
5	Interactive	40/week
6	Interactive	40/week

Access table for Operation 1

Concept	Construct	Accesses	Type
Match	Entity	1	R
Plays	Relationship	1	R
Team	Entity	2	R
Referee	Entity	40	R
ActiveMember	Entity	40	R
IsUnavailable	Relationship	80	R
Unavailability	Entity	80	R
Has	Relationship	50	R
MedicalCertificate	Entity	50	R
IsExempted	Relationship	40	R

Team	Entity	40	R
IsAssignedReferee	Relationship	200	R
RefereeAssignation	Entity	200	R
HasRefereeAssignation	Relationship	200	R
Match	Entity	200	

Access table for Operation 2

Concept	Construct	Accesses	Type
Match	Entity	1	R
Observer	Entity	10	R
ActiveMember	Relationship	10	R
IsUnavailable	Relationship	20	R
Unavailability	Entity	20	R
IsAssignedObserver	Relationship	50	R
ObserverAssignation	Entity	50	R
HasObserverAssignation	Relationship	50	R

Access table for Operation 3

Concept	Construct	Accesses	Type
IsAssignedReferee	Relationship	1	W
RefereeAssignation	Entity	1	W
HasRefereeAssignation	Relationship	1	W

Access table for Operation 4

Concept	Construct	Accesses	Type
IsAssignedObserver	Relationship	1	W
ObserverAssignation	Entity	1	W
HasObserverAssignation	Relationship	1	W

Access table for Operation 5

Concept	Construct	Accesses	Type

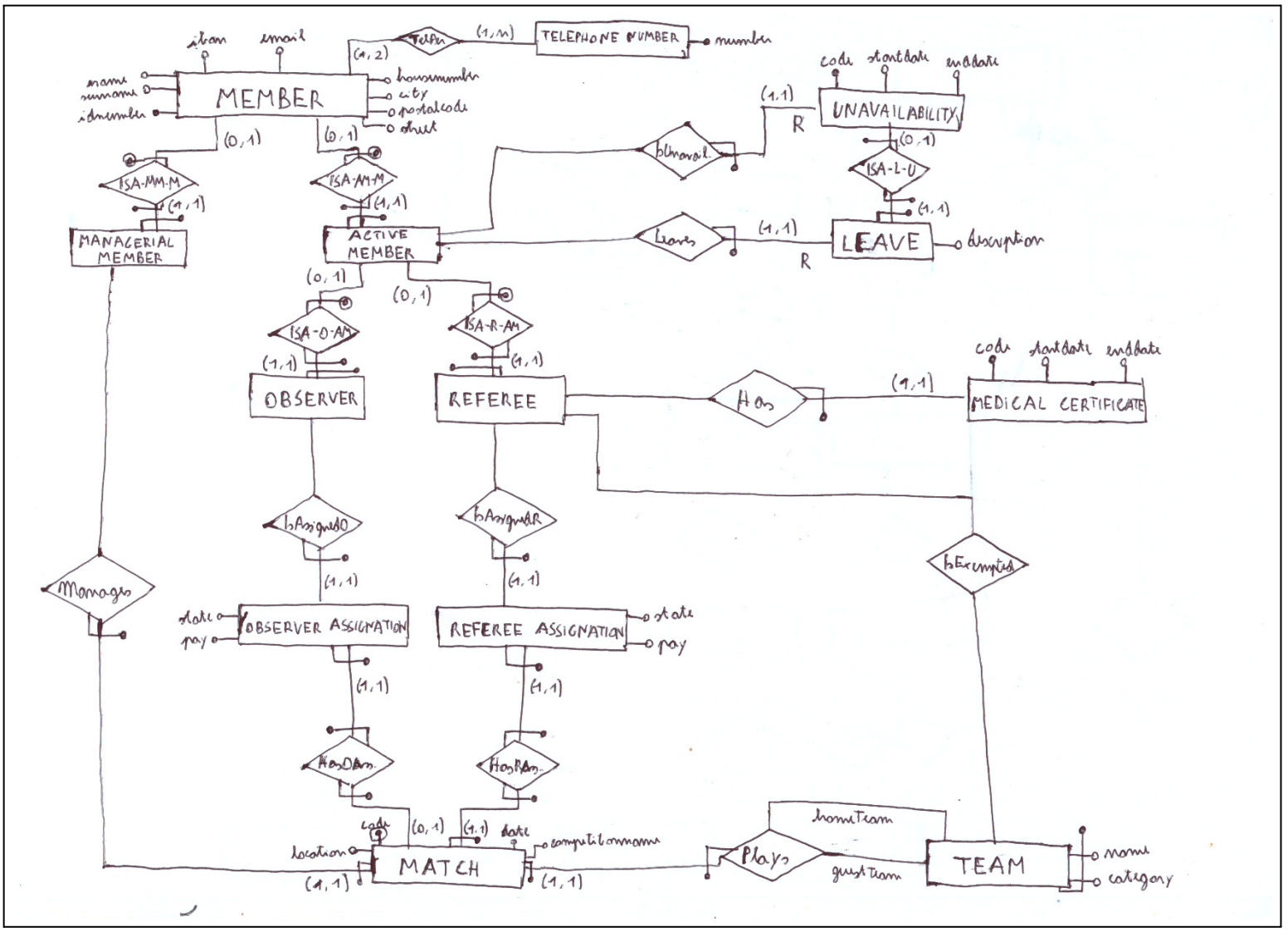
Assignation	Entity	1	W

Access table for Operation 6

Concept	**Construct**	**Accesses**	**Type**
Member	Entity	1	R

Restructuring of the conceptual schema

Restructured ER schema



External constraints:

1	*A referee can not be assigned to a match, if on the day of the match date the referee has no valid medical certificate.*
2	*A referee can not be assigned to a match, if on the day of the match date the referee is unavailable (is related to an unavailability).*
3	*A referee can not be assigned to a match that involves a team for which he asked to be exempted (not assigned).*

4	For each instance(R: e, ActiveMember: a) of “Leaves”. Let f be the instance of “Unavailability" such that (Leave: e, Unavailability: f) is an instance of “ISA-L-U”. Then (R: f, ActiveMember: a) is an instance of “IsUnavailable”.
5	Each instance of ActiveMember participates in ISA-O-AM or ISA-R-AM, but not in both.

Concept	**Construct**	**Volume**
Member	Entity	55
TelephoneNumber	Entity	80
ManagerialMember	Entity	1
Active member	Entity	50
Referee	Entity	40
Observer	Entity	10
Unavailability	Entity	100
Leave	Entity	2
MedicalCertificate	Entity	50
RefereeAssignation	Entity	200
ObserverAssignation	Entity	50
Match	Entity	200
Team	Entity	200
TelPer	Relationship	80
ISA-AM-M	Relationship	50
ISA-MM-M	Relationship	1
ISA-O-AM	Relationship	10
ISA-R-AM	Relationship	40
IsUnavailable	Relationship	100
Leaves	Relationship	2
ISA-L-U	Relationship	2
Has	Relationship	50
IsExempted	Relationship	40

Manages	Relationship	200
IsAssignedReferee	Relationship	200
IsAssignedObsever	Relationship	50
HasRefereeAssignation	Relationship	200
HasObserverAssignation	Relationship	50
Plays	Relationship	200

Access table for Operation 1

Concept	Construct	Accesses	Type
Match	Entity	1	R
Plays	Relationship	1	
Team	Entity	2	R
Referee	Entity	40	R
ISA-R-AM	Relationship	40	R
ActiveMember	Entity	40	R
IsUnavailable	Relationship	80	R
Unavailability	Entity	80	R
Has	Relationship	50	R
MedicalCertificate	Entity	50	R
IsExempted	Relationship	40	R
Team	Entity	40	R
IsAssignedReferee	Relationship	200	R
RefereeAssignation	Entity	200	R
HasRefereeAssignation	Relationship	200	R

Access table for Operation 2

Concept	Construct	Accesses	Type
Match	Entity	1	R
Observer	Entity	10	R
ISA-O-AC	Relationship	10	R

ActiveMember	Relationship	10	R
IsUnavailable	Relationship	20	R
Unavailability	Entity	20	R
IsAssignedObserver	Relationship	50	R
ObserverAssignation	Entity	50	R
HasObserverAssignation	Relationship	50	R

Access table for Operation 3

Concept	Construct	Accesses	Type
IsAssignedReferee	Relationship	1	W
RefereeAssignation	Entity	1	W
HasRefereeAssignation	Relationship	1	W

Access table for Operation 4

Concept	Construct	Accesses	Type
IsAssignedObserver	Relationship	1	W
ObserverAssignation	Entity	1	W
HasObserverAssignation	Relationship	1	W

Access table for Operation 5

Concept	Construct	Accesses	Type
Assignation	Entity	1	W

Access table for Operation 6

Concept	Construct	Accesses	Type
Member	Entity	55	R

(Direct) Relational schema

Member(idnumber, name, surname, iban, email, city, postalcode, street, housenumber)

inclusion: Member[idnumber] ⊆ TelPer[member]

TelPer(member, telephoneNumber)

foreign key: TelPer[member] ⊆ Member[idnumber]

foreign key: TelPer[telephoneNumber] ⊆ TelephoneNumber[number]

TelephoneNumber(number)

inclusion: TelephoneNumber[number] ⊆ TelPer[telephoneNumber]

ActiveMember(member)

foreign key: activeMember[member] ⊆ Member[idnumber]

ManagerialMember(member)

foreign key: ManagerialMember[member] ⊆ Member[idnumber]

Observer(activeMember)

foreign key: Observer[activeMember] ⊆ ActiveMember[member]

Referee(activeMember)

foreign key: Referee[activeMember] ⊆ ActiveMember[member]

IsUnavailable(unavailability, activeMember)

foreign key: IsUnavailable[unavailability] ⊆ Unavailability[code]

foreign key: IsUnavailable[activeMember] ⊆ ActiveMember[member]

Unavailability(code, startdate, enddate)

foreign key: Unavailability[code] ⊆ IsUnavailable[unavailability]

Leave(unavailability, description)

foreign key: Leave[unavailability] ⊆ Unavailability[code]

Leaves(leave, activeMember)

foreign key: Leaves[leave] ⊆ Leave[unavailability]

foreign key: Leaves[activeMember] ⊆ ActiveMember[member]

Has(medicalCertificate, referee)

foreign key: Has[medicalCertificate] ⊆ MedicalCertificate[code]

foreign key: Has[referee] ⊆ Referee[activeMember]

MedicalCertificate(code, startdate, enddate)

foreign key: MedicalCertificate[code] ⊆ Has[medicalCertificate]

Manages(match, managerialMember)

foreign key: Manages[match] ⊆ Match[code]

foreign key: Manages[managerialMember] ⊆ ManagerialMember[member]

Match(code, location, date, competitionName)

foreign key: Match[code] ⊆ Manages[match]

foreign key: Match[code] ⊆ Plays[match]

IsAssignedObserver(observerAssignation, observer)

foreign key: IsAssignedO[observerAssignation] ⊆ ObserverAssignation[match]

foreign key: IsAssignedO[observer] ⊆ Observer[activeMember]

ObserverAssignation(match, pay, state)

foreign key: ObserverAssignation[match] ⊆ IsAssignedO[observerAssignation]

foreign key: ObserverAssignation[match] ⊆ Match[code]

IsAssignedReferee(refereeAssignation, referee)

foreign key: IsAssignedR[refereeAssignation] ⊆ RefereeAssignation[match]

foreign key: IsAssignedR[referee] ⊆ Referee[activeMember]

RefereeAssignation(match, pay, state)

foreign key: RefereeAssignation[match] ⊆ IsAssignedR[refereeAssignation]

foreign key: RefereeAssignation[match] ⊆ Match[code]

IsExempted(referee, team, category)

foreign key: IsExempted[referee] ⊆ Referee[activeMember]

foreign key: IsExempted[team, category] ⊆ Team[team, category]

Team(name, category)

Plays(match, homeTeamName, homeTeamCategory, guestTeamName, guestTeamCategory)

foreign key: Plays[homeTeamName, homeTeamcategory] ⊆ Team[name, category]

foreign key: Plays[guestTeamName, guestTeamcategory] ⊆ Team[name, category]

1	*A referee can not be assigned to a match, if on the day of the match date the referee has no valid medical certificate.*
2	*A referee can not be assigned to a match, if on the day of the match date the referee is unavailable (is related to an unavailability).*
3	*A referee can not be assigned to a match that involves a team for which he is exempted.*
4	*For each instance(R: e, ActiveMember: a) of "Leaves". Let f be the instance of "Unavailability" such that (Leave: e, Unavailability: f) is an instance of "ISA-L-U". Then (R: f, ActiveMember: a) is an instance of "IsUnavailable".*
5	*Each instance of ActiveMember participates in ISA-O-AM or ISA-R-AM, but not in both.*

Access table for Operation 1

Concept	Accesses	Type
Match	1	R
Plays	1	R
Referee	40	R
Unavailability	80	R
IsUnavailable	80	R
Has	50	R
MedicalCertificate	50	R
IsAssignedReferee	200	R
RefereeAssignation	200	R
Match	200	R

Roughly description of the operation steps (for restructuring the relational schema):

Select a tuple of "Match" to then retrieve the date of that match.
Select the tuple of "Plays" that has as attribute "match" the code of the "Match" previously selected to retrieve the teams that will be playing.

a) *Select "Unavailability" tuples for which the date of the "Match" selected is not in the time period defined by "Unavailability" attributes: "startdate" and "enddate".*
Join the found "Unavailability" tuples with "IsUnavailable" ones and join again with "Referee" tuples.

b) *Select "MedicalCertificate" tuples which are valid on the date of the game. Join these "MedicalCertificate" tuples with "Has" tuples.*

c) *Select "IsExempted" tuples which have not as "Team" a team that will be playing in the match.*

d) *Join "RefereeAssignation" with "Match", select the tuples that have not as attribute "date" the date of the game and join the resulting tuples with "Referee" ones.*

Finally, join the tuples resulting after a, b, c, d to retrieve suitable referees for the match.

Access table for Operation 2

Concept	Accesses	Type
Match	1	R
Observer	10	R
Unavailability	20	R
IsUnavailable	20	R
IsAssignedObserver	50	R
ObserverAssignation	50	R
Match	50	R

Similar to Operation 1.
Select a tuple of match.

a) *Select "Unavailability" tuples for which the date of the "Match" selected is not in the time period defined by "Unavailability" attributes: "startdate" and "enddate". Join the found "Unavailability" tuples with "IsUnavailable" ones and join again with "Observer" tuples.*

b) *Join "ObserverAssignation" with "Match", select then the tuples that have not as attribute "date" the date of the game and finally join those tuples with "Observer" ones.*

Finally join tuples resulting from a and b to retrieve suitable observers for a certain match.

Access table for Operation 3

Concept	Accesses	Type
IsAssignedReferee	1	W
RefereeAssignation	1	W

Inserting a new "IsAssignedReferee" and "RefereeAssignation" tuple.

Access table for Operation 4

Concept	Accesses	Type
IsAssignedObserver	1	W
ObserverAssignation	1	W

Inserting a new "IsAssignedObserver" and "ObserverAssignation" tuple.

Access table for Operation 5

Concept	Accesses	Type
Assignation	1	W

Inserting a new "Assignation" tuple.

Access table for Operation 6

Concept	Accesses	Type
Member	1	R

Selecting a certain member.

Intuitively we realize that there are two types of inefficiencies.

1) The first one refers to the relation "Member", because it has two sets of attributes which are accessed separately during operation 6. Therefore we are going to apply vertical decomposition.
2) The second one refers instead to the several joins which are performed in operation 1 and 2. For that reason we are going to merge some relations to facilitate access.

1)

before:

Member(idnumber, name, surname, iban, email, city, postalcode, street, housenumber)

inclusion: Member[idnumber] ⊆ TelPer[member]

after:

MemberGen(idnumber, name, surname, iban, email)

foreign key: MemberGen[idnumber] ⊆ MemberLoc[idnumber]

inclusion: Member[idnumber] ⊆ TelPer[member]

MemberLoc(idnumber, city, postalcode, street, housenumber)

foreign key: MemberLoc[idnumber] ⊆ MemberGen[idnumber]

inclusion: Member[idnumber] ⊆ TelPer[member]

2)

before:

IsUnavailable(unavailability, activeMember)

foreign key: IsUnavailable[unavailability] ⊆ Unavailability[code]

foreign key: IsUnavailable[activeMember] ⊆ ActiveMember[member]

Unavailability(code, startdate, enddate)

foreign key: Unavailability[code] ⊆ IsUnavailable[unavailability]

after:

Unavailability(code, startdate, enddate, activeMember)

foreign key: Unavailability[code] ⊆ ActiveMember[member]

---

before:

Has(medicalCertificate, referee)

foreign key: Has[medicalCertificate] ⊆ MedicalCertificate[code]

foreign key: Has[referee] ⊆ Referee[activeMember]

MedicalCertificate(code, startdate, enddate)

foreign key: MedicalCertificate[code] ⊆ Has[medicalCertificate]

after:

MedicalCertificate(code, startdate, enddate, referee)

foreign key: MedicalCertificate[code] ⊆ Referee[activeMember]

---

before:

IsAssignedObserver(observerAssignation, observer)

foreign key: IsAssignedO[observerAssignation] ⊆ ObserverAssignation[match]

foreign key: IsAssignedO[observer] ⊆ Observer[activeMember]

ObserverAssignation(match, pay, state)

foreign key: ObserverAssignation[match] ⊆ IsAssignedO[observerAssignation]

foreign key: ObserverAssignation[match] ⊆ Match[code]

after:

ObserverAssignation(match, pay, state, observer)

foreign key: ObserverAssignation[observer] ⊆ Observer[activeMember]

foreign key: ObserverAssignation[match] ⊆ Match[code]

---

before:

IsAssignedReferee(refereeAssignation, referee)

foreign key: IsAssignedR[refereeAssignation] ⊆ RefereeAssignation[match]

foreign key: IsAssignedR[referee] ⊆ Referee[activeMember]

RefereeAssignation(match, pay, state)

foreign key: RefereeAssignation[match] ⊆ IsAssignedR[refereeAssignation]

foreign key: RefereeAssignation[match] ⊆ Match[code]

after:

RefereeAssignation(match, pay, state, referee)

foreign key: RefereeAssignation[referee] ⊆ Referee[activeMember]

foreign key: RefereeAssignation[match] ⊆ Match[code]

Restructured relational schema

MemberGen(idnumber, name, surname, iban, email)

foreign key: MemberGen[idnumber] ⊆ MemberLoc[idnumber]

inclusion: Member[idnumber] ⊆ TelPer[member]

MemberLoc(idnumber, city, postalcode, street, housenumber)

foreign key: MemberLoc[idnumber] ⊆ MemberGen[idnumber]

inclusion: Member[idnumber] ⊆ TelPer[member]

TelPer(member, telephoneNumber)

foreign key: TelPer[member] ⊆ Member[idnumber]

foreign key: TelPer[telephoneNumber] ⊆ TelephoneNumber[number]

TelephoneNumber(number)

inclusion: TelephoneNumber[number] ⊆ TelPer[telephoneNumber]

ActiveMember(member)

foreign key: activeMember[member] ⊆ Member[idnumber]

ManagerialMember(member)

foreign key: ManagerialMember[member] ⊆ Member[idnumber]

Observer(activeMember)

foreign key: Observer[activeMember] ⊆ ActiveMember[member]

Referee(activeMember)

foreign key: Referee[activeMember] ⊆ ActiveMember[member]

Unavailability(code, startdate, enddate, activeMember)

foreign key: Unavailability[code] ⊆ ActiveMember[member]

Leave(unavailability, description)

foreign key: Leave[unavailability] ⊆ Unavailability[code]

Leaves(leave, activeMember)

foreign key: Leaves[leave] ⊆ Leave[unavailability]

foreign key: Leaves[activeMember] ⊆ ActiveMember[member]

MedicalCertificate(code, startdate, enddate, referee)

foreign key: MedicalCertificate[code] ⊆ Referee[activeMember]

ObserverAssignation(match, pay, state, observer)

foreign key: ObserverAssignation[observer] ⊆ Observer[activeMember]

foreign key ObserverAssignation[match] ⊆ Match[code]

RefereeAssignation(match, pay, state, referee)

foreign key: RefereeAssignation[referee] ⊆ Referee[activeMember]

foreign key RefereeAssignation[match] ⊆ Match[code]

Manages(match, managerialMember)

foreign key: Manages[match] ⊆ Match[code]

foreign key: Manages[managerialMember] ⊆ ManagerialMember[member]

Match(code, location, date, competitionName)

foreign key: Match[code] ⊆ Manages[match]

foreign key: Match[code] ⊆ Plays[match]

IsExempted(referee, team, category)

foreign key: IsExempted[referee] ⊆ Referee[activeMember]

foreign key: IsExempted[team, category] ⊆ Team[team, category]

Team(name, category)

Plays(match, homeTeamName, homeTeamCategory, guestTeamName, guestTeamCategory)

foreign key: Plays[homeTeamName, homeTeamcategory] ⊆ Team[name, category]

foreign key: Plays[guestTeamName, guestTeamcategory] ⊆ Team[name, category]

Specification of the Database in SQL

```
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)
);
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)
);
CREATE TABLE TelPer (
        member INT,
        telephoneNumber VARCHAR(15),
        PRIMARY KEY (member, telephoneNumber),
        FOREIGN KEY (member) REFERENCES MemberGen(idnumber),
        FOREIGN KEY (telephoneNumber) REFERENCES TelephoneNumber(number)
);
 CREATE TABLE TelephoneNumber (
        number VARCHAR(15) PRIMARY KEY
```

```
);
 CREATE TABLE ActiveMember (
        member INT PRIMARY KEY,
        FOREIGN KEY (member) REFERENCES MemberGen(idnumber)
);
 CREATE TABLE ManagerialMember (
        member INT PRIMARY KEY,
        FOREIGN KEY (member) REFERENCES MemberGen(idnumber)
);
 CREATE TABLE Observer (
        activeMember INT PRIMARY KEY,
        FOREIGN KEY (activeMember) REFERENCES ActiveMember(member)
);
 CREATE TABLE Referee (
        activeMember INT PRIMARY KEY,
        FOREIGN KEY (activeMember) REFERENCES ActiveMember(member)
);
 CREATE TABLE Unavailability (
        code INT PRIMARY KEY,
        startdate DATE,
        enddate DATE,
        activeMember INT,
        FOREIGN KEY (activeMember) REFERENCES ActiveMember(member)
);
```

```
CREATE TABLE Leave (
    unavailability INT PRIMARY KEY,
        description TEXT,
        FOREIGN KEY (unavailability) REFERENCES Unavailability(code)
);
 CREATE TABLE Leaves (
        leave INT,
        activeMember INT,
        PRIMARY KEY (leave, activeMember),
        FOREIGN KEY (leave) REFERENCES Leave(unavailability),
        FOREIGN KEY (activeMember) REFERENCES ActiveMember(member)
);
 CREATE TABLE MedicalCertificate (
        code INT PRIMARY KEY,
        startdate DATE,
        enddate DATE,
        referee INT,
        FOREIGN KEY (referee) REFERENCES Referee(activeMember)
);
 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)
);
 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)
);
 CREATE TABLE Manages (
        match INT,
        managerialMember INT,
        PRIMARY KEY (match, managerialMember),
        FOREIGN KEY (match) REFERENCES Match(code),
        FOREIGN KEY (managerialMember) REFERENCES ManagerialMember(member)
);
 CREATE TABLE Match (
        code INT PRIMARY KEY,
        location VARCHAR(100),
        date DATE,
    competitionName VARCHAR(100)
);
```

```
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)
);
 CREATE TABLE Team (
        name VARCHAR(100),
        category VARCHAR(50),
        PRIMARY KEY (name, category)
);
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)
);
```

```
CREATE OR REPLACE FUNCTION check_referee_medical_certificate() RETURNS
TRIGGER AS $$

BEGIN

	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();


CREATE OR REPLACE FUNCTION check_referee_unavailability() RETURNS TRIGGER
AS $$

BEGIN

	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();

CREATE OR REPLACE FUNCTION check_referee_exemption() RETURNS TRIGGER AS
$$
BEGIN
        IF EXISTS (
        SELECT 1 FROM IsExempted
        WHERE referee = NEW.referee
        AND (team = (SELECT homeTeamName FROM Plays WHERE match =
NEW.match)
        OR team = (SELECT guestTeamName FROM Plays WHERE match = NEW.match))
        ) THEN
        RAISE EXCEPTION '';
        END IF;
        RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_check_referee_exemption
```

```
BEFORE INSERT OR UPDATE ON RefereeAssignation

FOR EACH ROW EXECUTE FUNCTION check_referee_exemption();
```