GEBZE TECHNICAL UNIVERSITY
SPRING 2021
COMPUTER ENGINEERING
CSE414 – DATABASE
PROJECT REPORT

BASESOCIAL DATABASE MANAGEMENT SYSTEM

GÖKHAN HAS – 161044067

LECTURER : DR. BURCU YILMAZ

1. WHAT IS BASESOCIAL?

My project is a web application designed for baseball lovers around the world. Supporters can share their own ideas about teams and organizations. They can also view the rosters, managers, and managers of their baseball club. After the contract renewals are notified to the relevant federation, the application falls on it. Likewise, the latest status in the leagues, match statistics, transfer status, transfer requests, etc. A lot of extra information is also kept. Supporters can make recommendations to their team, other teams, or themselves.

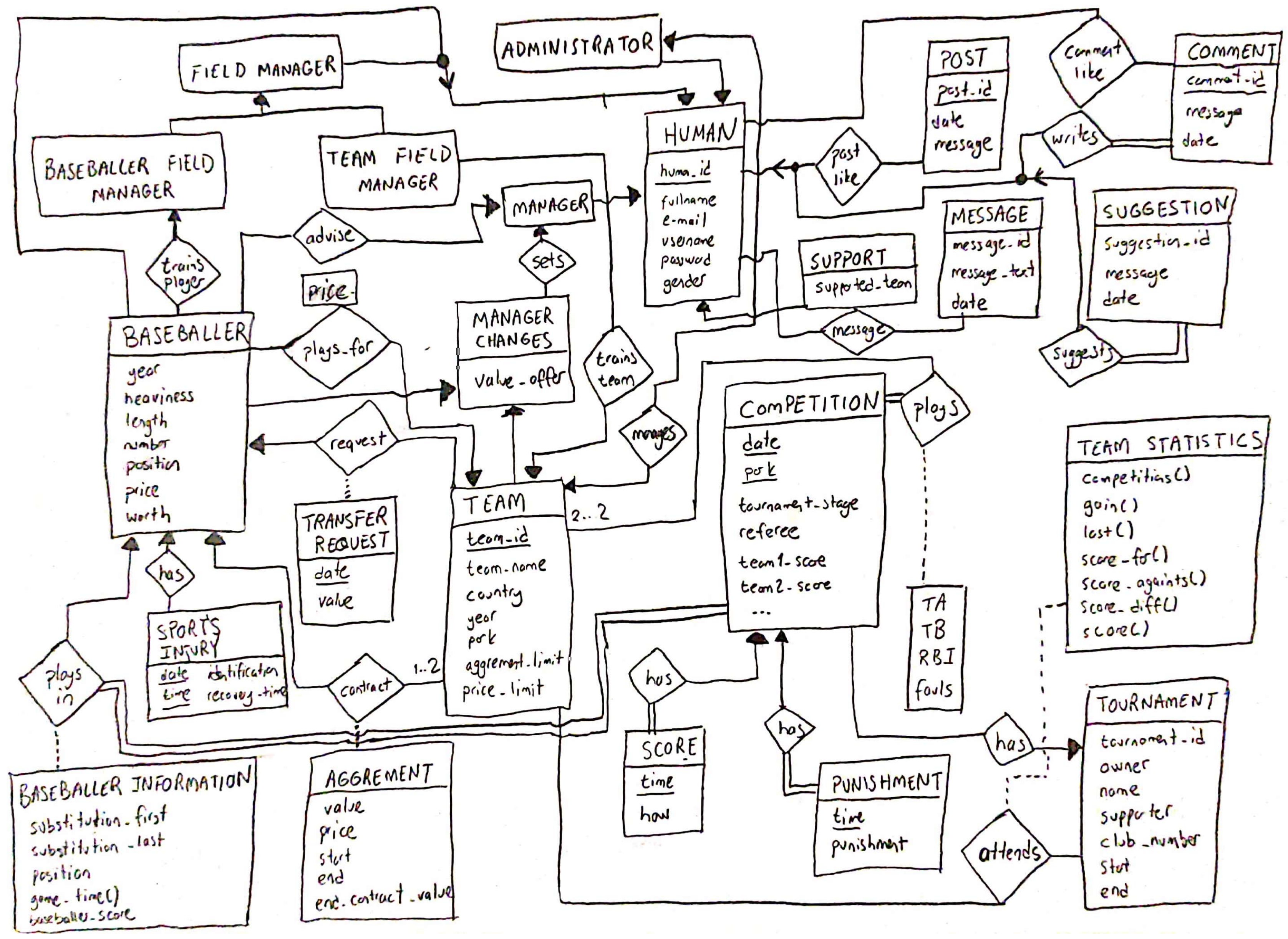
2. USER REQUIREMENTS

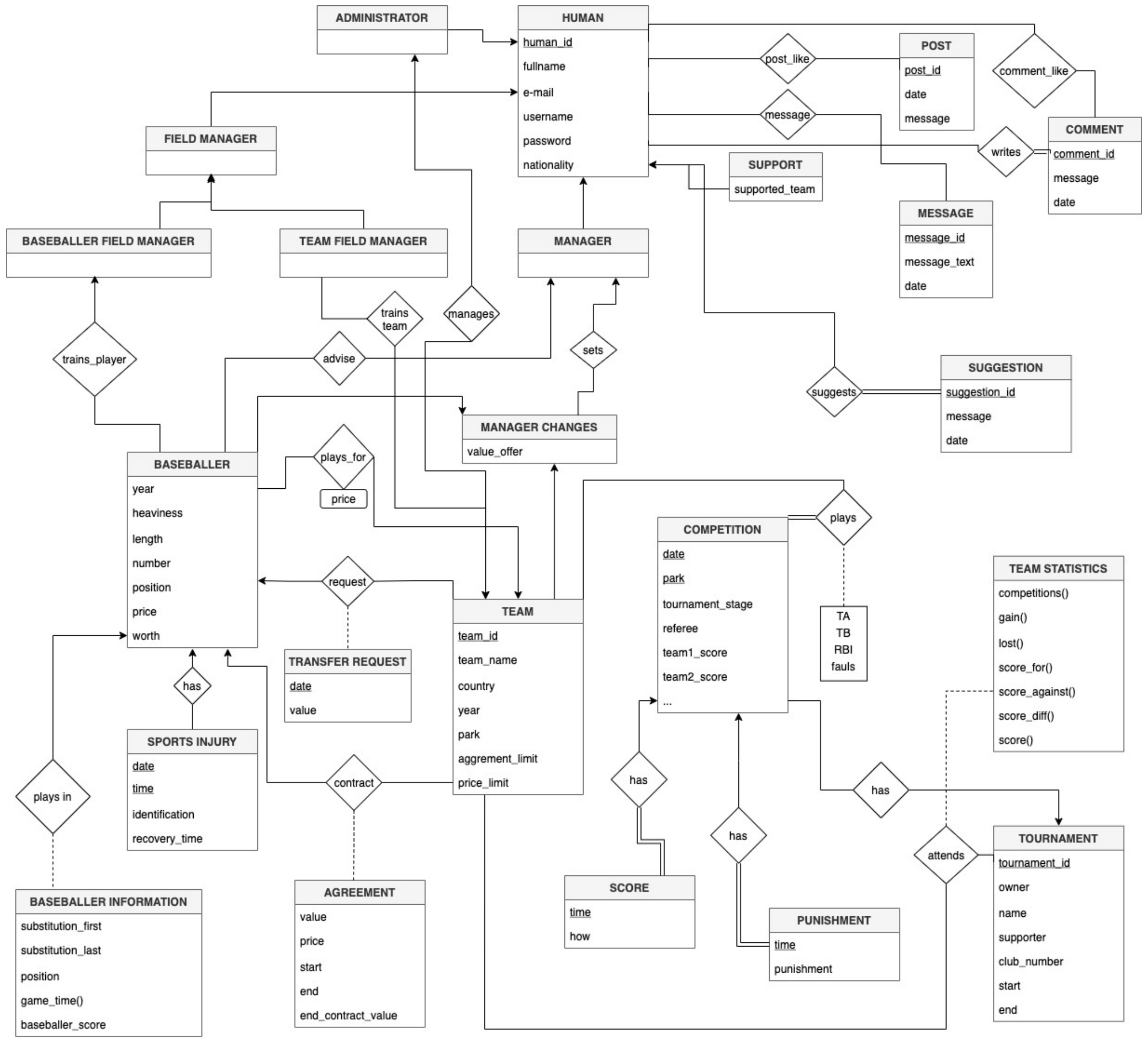
This web application has five types of end users. These are divided into two: one is the supporters and the other is the people involved in baseball. Baseball promoters have relatively little authority among other genres. However, its effects are high in terms of potency. They can choose their favorite baseball teams and players if any. They should be able to receive all the information about their favorite team as notification. Likewise, they should be able to get information about events. Organizers represent confederations such as the WBSC. It should be able to create new activities linked to their confederation. Club Managers are team managers who make decisions about teams and baseballers. She/he should be able to put baseballers on the transfer list by determining their values. Must be able to offer transfers for baseballers from other teams. She/He must be able to reject transfer offers from other teams. Team field managers organize team tactics and take care of the baseballers. Must be able to select team captains and adjust formation tactics. Managers are the personal managers of the baseballers, they should be able to accept or reject the offers from the teams, and see the offers related to the baseballers they manage.

Access permissions can only be changed by the system administrator. Each type of user can log in but access different levels, so the permissions and limitations of the user's actions must be specified. length etc for safety. password requirements. The web application should be user friendly, everything in the system should be open for the user to use. It should also be fast. It shouldn't fail. It must continue to perform its necessary functions under the specified conditions. Since this system is a database system, it may need to store a huge amount of data due to the number of teams, baseballers, leagues and seasons. This should be noted.

3. ER DIAGRAM

If you want to examine the diagram in more detail, it is also included in the zip as a picture.


FIELD MANAGER
ADMINISTRATOR
POST
post_id
date
message
comment like
COMMENT
comment_id
message
date
HUMAN
human_id
fullname
e-mail
username
password
gender
post like
writes
BASEBALLER FIELD MANAGER
TEAM FIELD MANAGER
MANAGER
advise
trains player
sets
price
MESSAGE
message_id
message_text
date
SUGGESTION
suggestion_id
message
date
SUPPORT
supported_team
message
suggests
BASEBALLER
year
heaviness
length
number
position
price
worth
plays_for
MANAGER CHANGES
value_offer
trains team
manages
COMPETITION
date
park
tournament_stage
referee
team1_score
team2_score
...
plays
request
TRANSFER REQUEST
date
value
TEAM
team_id
team_name
country
year
park
aggrement_limit
price_limit
2...2
TEAM STATISTICS
competitions()
gain()
lost()
score_for()
score_againts()
score_diff()
score()
TA
TB
RBI
fouls
has
SPORTS INJURY
date
identification
time
recovery_time
plays in
contract
1..2
has
has
SCORE
time
how
PUNISHMENT
time
punishment
has
TOURNAMENT
tournament_id
owner
name
supporter
club_number
start
end
attends
BASEBALLER INFORMATION
substitution_first
substitution_last
position
game_time()
baseballer_score
AGGREMENT
value
price
start
end
end_contract_value


ADMINISTRATOR
HUMAN
human_id
fullname
e-mail
username
password
nationality
post_like
POST
post_id
date
message
comment_like
message
COMMENT
comment_id
message
date
writes
SUPPORT
supported_team
MESSAGE
message_id
message_text
date
FIELD MANAGER
BASEBALLER FIELD MANAGER
TEAM FIELD MANAGER
MANAGER
trains_player
trains team
manages
advise
sets
suggests
SUGGESTION
suggestion_id
message
date
MANAGER CHANGES
value_offer
BASEBALLER
year
heaviness
length
number
position
price
worth
plays_for
price
request
TRANSFER REQUEST
date
value
TEAM
team_id
team_name
country
year
park
aggrement_limit
price_limit
plays
COMPETITION
date
park
tournament_stage
referee
team1_score
team2_score
...
TA
TB
RBI
fauls
TEAM STATISTICS
competitions()
gain()
lost()
score_for()
score_against()
score_diff()
score()
has
SPORTS INJURY
date
time
identification
recovery_time
plays in
BASEBALLER INFORMATION
substitution_first
substitution_last
position
game_time()
baseballer_score
contract
AGREEMENT
value
price
start
end
end_contract_value
has
SCORE
time
how
has
PUNISHMENT
time
punishment
has
attends
TOURNAMENT
tournament_id
owner
name
supporter
club_number
start
end

4. FUNCTIONAL DEPENDENCIES AND TABLES

4.1. Human

- **Relation Schema:**

  Human(<u>human_id</u>, fullname, e-mail, username, password, gender)
- **Function Dependencies:**

human_id	→ fullname, e-mail, username, password, gender)
username	→ human_id, fullname, e-mail, password, gender)
e-mail	→ human_id, fullname, username, password, gender)

- **Primary Key** : {(human_id)}
- **Candidate Keys** : {(human_id), (username), (e-mail)}
- **Normal Form** : BCNF
- **Table** :

```
CREATE TABLE Human(
        human_id INT PRIMARY KEY AUTO_INCREMENT,
        fullname VARCHAR(40) NOT NULL,
        e-mail VARCHAR(40) NOT NULL UNIQUE,
        username VARCHAR(40) NOT NULL UNIQUE,
        password VARCHAR(40) NOT NULL,
        gender VARCHAR(40)) ENGINE = InnoDB;
```

4.2. Baseballer

- **Relation Schema** :

  Baseballer(<u>baseballer_id</u>, year, heaviness, length, number, club, position, manager, field_manager, price, worth)
- **Function Dependencies** :

  baseballer_id → year, heaviness, length, club, position, number, manager, field_manager, price, worth

  club, number → baseballer_id, year, heaviness, length, position, manager, field_manager, wage, value
- **Primary Key** : {(baseballer_id)}
- **Candidate Keys** : {(baseballer_id), (club, number)}
- **Normal Form** : BCNF
- **Table** :

```
CREATE TABLE Baseballer(
        baseballer_id INT,
        year MEDIUMINT,
        heaviness TINYINT,
        length TINYINT,
        number TINYINT,
        club INT,
        position ENUM('LF', 'CF', 'RF', 'SS', '2B', '3B', '1B', 'P', 'C' ),
        manager INT,
        field_manager INT,
        price INT,
        worth INT,
        UNIQUE (club, number),
        FOREIGN KEY (baseballer_id) REFERENCES Human(human_id),
        FOREIGN KEY (club) REFERENCES Team(team_id),
        FOREIGN KEY (manager) REFERENCES Manager(manager_id),
        FOREIGN KEY (field_manager) REFERENCES
        FieldManager(field_manager_id)) ENGINE = InnoDB;
```

4.3. Transfer Request

- **Relation Schema:**
  TransferRequest(<u>baseballer, club_sell, club_buy</u>, date, value)
- **Function Dependencies:**
  baseballer, club_sell, club_buy, date → value
- **Primary Key** : {(baseballer, club_sell, club_buy, date)}
- **Candidate Keys** : {(baseballer, club_sell, club_buy, date)}
- **Normal Form** : BCNF
- **Table** :

```
CREATE TABLE Offer(
        baseballer INT,
        club_sell INT,
        club_buy INT,
        date DATE,
        value INT,
        PRIMARY KEY (baseballer, club_sell, club_buy, date),
        FOREIGN KEY (baseballer) REFERENCES Baseballer(baseballer_id),
        FOREIGN KEY (club_sell) REFERENCES Team(team_id),
        FOREIGN KEY (club_buy) REFERENCES Team(team_id))
        ENGINE = InnoDB;
```

4.4. ManagerChanges

- **Relation Schema:**
  ManagerChanges(<u>baseballer, club_sell, club_buy, date</u>, value_offer)
- **Function Dependencies:**
  baseballer, club_sell, club_buy, date → value_offer
- **Primary Key** : {(baseballer, club_sell, club_buy, date)}
- **Candidate Keys** : {(baseballer, club_sell, club_buy, date)}
- **Normal Form** : BCNF
- **Table** :

```
CREATE TABLE ManagerSet(
        baseballer INT,
        club_sell INT,
        club_buy INT,
        date DATE,
        value_offer INT,
        PRIMARY KEY (baseballer, club_sell, club_buy, date),
        FOREIGN KEY (baseballer, club_sell, club_buy, date) REFERENCES
        TransferRequest) ENGINE = InnoDB;
```

4.5. Support

- **Relation Schema:**

  Support(*support_id*, supported_team)
- **Function Dependencies:**

  support_id → supported_team
- **Primary Key** : {(support_id)}
- **Candidate Keys** : {(support_id)}
- **Normal Form** : BCNF
- **Table** :

```
CREATE TABLE Fan(
        support_id INT PRIMARY KEY,
        supported_team INT,
        FOREIGN KEY (fan_id) REFERENCES Human(human_id),
        FOREIGN KEY (supported_team) REFERENCES Team(team_id))
ENGINE = InnoDB;
```

4.6. Baseballer Field Manager

- **Relation Schema:**
  BaseballerFieldManager(baseballer_field_manager_id)
- **Function Dependencies:** No FD
- **Primary Key** : {(baseballer_field_manager_id)}
- **Candidate Keys** : {(baseballer_field_manager_id)}
- **Normal Form** : BCNF
- **Table** :

```
CREATE TABLE BaseballerFieldManager (
        baseballer_field_manager_id INT PRIMARY KEY,
        FOREIGN KEY (baseballer_field_manager_id) REFERENCES
        Human(human_id)) ENGINE = InnoDB;
```

4.7. Team Field Manager

- **Relation Schema:**
  TeamFieldManager(team_field_manager_id)
- **Function Dependencies:** No FD
- **Primary Key** : {(team_field_manager_id)}
- **Candidate Keys** : {(team_field_manager_id)}
- **Normal Form** : BCNF
- **Table** :

```
CREATE TABLE TeamFieldManager (
        team_field_manager_id INT PRIMARY KEY,
        FOREIGN KEY (team_field_manager_id) REFERENCES
        Human(human_id)) ENGINE = InnoDB;
```

4.8. Administrator

- **Relation Schema:**
  Administrator(admin_id)
- **Function Dependencies:** No FD
- **Primary Key** : {(admin_id)}
- **Candidate Keys** : {(admin_id)}
- **Normal Form** : BCNF
- **Table** :

```
CREATE TABLE Administrator (
        admin_id INT PRIMARY KEY,
        FOREIGN KEY (admin_id) REFERENCES Human(human_id))
        ENGINE = InnoDB;
```

4.9. Score

- **Relation Schema** :
  Score(date, park, time, baseballer, how)
- **Function Dependencies** :
  date, park, time → baseballer, how
- **Primary Key** : {(date, park, time)}
- **Candidate Keys** : {(date, park, time)}
- **Normal Form** : BCNF
- **Table** :

```
CREATE TABLE Score(
        date DATETIME,
        park VARCHAR(40),
        time TIME,
        baseballer INT,
        how VARCHAR(40),
        PRIMARY KEY (date, park, time),
        FOREIGN KEY (date, park) REFERENCES Competition,
        FOREIGN KEY (baseballer) REFERENCES Baseballer(baseballer _id)
        ENGINE = InnoDB;
```

4.10. Sports Injury

- **Relation Schema:**
  SportsInjury(baseballer_id, date, time, identification, recovery_time)
- **Function Dependencies:**
  baseballer_id, date, time → identification, recovery_time
- **Primary Key** : {(baseballer_id, date, time)}

- **Candidate Keys** : {(baseballer_id, date, time)}
- **Normal Form** : BCNF
- **Table** :

```
CREATE TABLE SportsInjury(
        baseballer_id INT,
        date DATETIME,
        time TIME,
        identification VARCHAR(40),
        recovery_time CHAR(10),
        PRIMARY KEY (baseballer_id, date, time),
        FOREIGN KEY (baseballer_id) REFERENCES Baseballer,
        FOREIGN KEY (date) REFERENCES Competition(date)) ENGINE =
        InnoDB;
```

4.11. Manager

- **Relation Schema:**
  Manager(<u>manager_id</u>)
- **Function Dependencies:** No FD
- **Primary Key** : {(manager_id)}
- **Candidate Keys** : {(manager_id)}
- **Normal Form** : BCNF
- **Table** :

```
CREATE TABLE Manager(
        manager_id INT PRIMARY KEY,
        FOREIGN KEY (manager_id) REFERENCES Human(human_id))
        ENGINE = InnoDB;
```

4.12. Field Manager

- **Relation Schema** :
  FieldManager(<u>field_manager_id</u>)
- **Function Dependencies:** No FD
- **Primary Key** : {(field_manager_id)}
- **Candidate Keys** : {(field_manager_id)}
- **Normal Form** : BCNF
- **Table** :

```
CREATE TABLE FieldManager(
        field_manager_id INT PRIMARY KEY,
        FOREIGN KEY (field_manager_id) REFERENCES
        Human(human_id)) ENGINE = InnoDB;
```

4.13. Agreement

- **Relation Schema:**
  Agreement(baseballer, club_sell, club_buy, start, value, price, end, end_contract_value)
- **Function Dependencies:**
  baseballer, club_sell, club_buy, start → value, price, end, end_contract_value
  baseballer, club_sell, club_buy, end → start, value, price, end_contract_value
- **Primary Key** : {(baseballer, club_sell, club_buy, start)}
- **Candidate Keys** :
  {(baseballer, club_sell, club_buy, start) , (baseballer, club_sell, club_buy, end)}
- **Normal Form** : BCNF
- **Table** :

```
CREATE TABLE Agreement(
        baseballer INT,
        club_sell INT,
        club_buy INT,
        start DATE,
        end DATE,
        value INT,
        priceINT,
        end_conract_value INT,
        PRIMARY KEY (baseballer, club_sell, club_buy, start),
        UNIQUE (baseballer, club_sell, club_buy, end
        FOREIGN KEY (baseballer) REFERENCES Baseballer(baseballer_id),
        FOREIGN KEY (club_sell) REFERENCES Team(team_id),
        FOREIGN KEY (club_buy) REFERENCES Team(team_id))
        ENGINE = InnoDB;
```

4.14. Team

- **Relation Schema:**
  Team(team_id, team_name, country, year, park, director, field_manager, aggrement_limit, price_limit)
- **Function Dependencies:**
  team_id → team_name, country, year, park, director, field_manager, aggrement_limit, price_limit
- **Primary Key** : {(team_id)}
- **Candidate Keys** : {(team_id)}
- **Normal Form** : BCNF
- **Table** :

```
CREATE TABLE Team(
        team_id INT PRIMARY KEY,
        team_name VARCHAR(40) NOT NULL,
        country VARCHAR(40) NOT NULL,
        year DATE NOT NULL,
        park VARCHAR(40),
        admin VARCHAR(40),
        field_manager VARCHAR(40),
        aggrement_limit INT,
        price_limit INT,
        FOREIGN KEY (admin) REFERENCES
        Administrator(admin_id),
        FOREIGN KEY (field_manager) REFERENCES
        FieldManager(field_manager_id))
        ENGINE = InnoDB;
```

4.15. Team Statistics

- **Relation Schema:**
  TeamStatistics(<u>tournament_id, team_id</u>, competitions(), gain(), lost(), score_for(), score_against(), score_diff(), score())
- **Function Dependencies:**
  tournament_id, team_id → competitions(), gain(), lost(), score_for(), score_against(), score_diff(), score()
- **Primary Key** : {(tournament_id, team_id)}
- **Candidate Keys** : {(tournament_id team_id)}
- **Normal Form** : BCNF
- **Table** :

```
CREATE TABLE TeamStatistics(
        tournament_id INT,
        team_id INT,
        competitions TINYINT,
        gain TINYINT,
        lost TINYINT,
        score_for TINYINT,
        score_against TINYINT,
        score_diff TINYINT,
        score TINYINT,
        PRIMARY KEY (tournament_id, team_id),
        FOREIGN KEY (tournament_id) REFERENCES Tournament,
        FOREIGN KEY (team_id) REFERENCES Team) ENGINE = InnoDB;
```

4.16. Competition

- **Relation Schema:**
  Competition (<u>date, park</u>, tournament, tournament_stage, team1, team2, referee, team1_score, team2_score, team1_TA, team2_TA, team1_TB, team2_TB, team1_RBI, team2_RBI, team1_fauls, team2_fauls)
- **Function Dependencies:**
  date, park → tournament, tournament _stage, team1, team2, referee, team1_score, team2_score, team1_TA, team2_TA, team1_TB, team2_TB, team1_RBI, team2_RBI, team1_fauls, team2_fauls
  date, team1, team2 → park, tournament, torunament_stage, referee, team1_score, team2_score, team1_TA, team2_TA, team1_TB, team2_TB, team1_RBI, team2_RBI, team1_fauls, team2_fauls

- **Primary Key** : {(date, park)}
- **Candidate Keys** : {(date, park), (date, team1, team2)}
- **Normal Form** : BCNF
- **Table** :

```
CREATE TABLE Competition(
        date DATETIME,
        park VARCHAR(40),
        tournament VARCHAR(40),
        tournament_stage VARCHAR(40),
        team1 INT,
        team2 INT,
        referee VARCHAR(40),
        team1_score TINYINT,
        team2_score TINYINT,
        team1_TA TINYINT,
        team2_TA TINYINT,
        team1_TB TINYINT,
        team2_TB TINYINT,
        team1_RBI TINYINT,
        team2_RBI TINYINT,
        team1_fauls TINYINT,
        team2_fauls TINYINT,
        PRIMARY KEY (date, park),
        UNIQUE (date, team1, team2),
        FOREIGN KEY (park) REFERENCES Team(park),
        FOREIGN KEY (team1) REFERENCES Team(team_id),
        FOREIGN KEY (team2) REFERENCES Team(team_id))
        ENGINE = InnoDB;
```

4.17. Tournament

- **Relation Schema:**
  Tournament(tournament_id, owner, name, supporter, club_number(), start, end)
- **Function Dependencies:**
  tournament_id → organizer, name, supporter, club_number(), start, end
- **Primary Key** : {(tournament_id)}
- **Candidate Keys** : {(tournament_id)}
- **Normal Form** : BCNF
- **Table** :

```
CREATE TABLE Tournament(
        tournament_id INT PRIMARY KEY,
        owner VARCHAR(40) NOT NULL,
        name VARCHAR(40) NOT NULL,
        supporter VARCHAR(40),
        club_number TINYINT,
        start DATE,
        end DATE);
```

4.18. Post

- **Relation Schema:**
  Post(<u>post_id</u>, date, message, support_id)
- **Function Dependencies:**
  post_id → date, message, support_id
- **Primary Key** : {(post_id)}
- **Candidate Keys** : {(post_id)}
- **Normal Form** : BCNF
- **Table** :

```
CREATE TABLE Post (
        post_id INT NOT NULL,
        date DATE NOT NULL,
        message VARCHAR(255),
        support_id INT NOT NULL,
        FOREIGN KEY(support_id) REFERENCES Human(human_id),
        PRIMARY KEY (post_id)) ENGINE = InnoDB;
```

4.19. Post Like

- **Relation Schema:**
  PostLike(<u>support_id, post_id</u>)
- **Function Dependencies:** No FD
- **Primary Key** : {(support_id, post_id)}
- **Candidate Keys** : {(support_id, post_id)}
- **Normal Form** : BCNF
- **Table** :

```
CREATE TABLE PostLike (
        post_id INT NOT NULL,
        support_id INT NOT NULL,
        FOREIGN KEY(support_id) REFERENCES Human(human_id),
        FOREIGN KEY(post_id) REFERENCES Post(post_id),
        PRIMARY KEY (post_id, support_id)) ENGINE = InnoDB;
```

4.20. Comment

- **Relation Schema:**
  Comment(<u>post_id, comment_id</u>, message, date, fan_id)
- **Function Dependencies:**
  post_id, comment_id → message, date, fan_id
- **Primary Key** : {(post_id, comment_id)}
- **Candidate Keys** : {(post_id, comment_id)}
- **Normal Form** : BCNF
- **Table** :

```
CREATE TABLE Comment (
        post_id INT NOT NULL,
        comment_id INT AUTO_INCREMENT,
        message VARCHAR(255) NOT NULL,
        date DATE NOT NULL,
        fan_id INT NOT NULL,
        FOREIGN KEY(post_id) REFERENCES Post(post_id),
        FOREIGN KEY(fan_id) REFERENCES Human(human_id),
        PRIMARY KEY (post_id, comment_id)) ENGINE = InnoDB;
```

4.21. Suggestion

- **Relation Schema:**
  Suggestion(<u>suggestion_id,</u> message, date)
- **Function Dependencies:**
  suggestion_id → message, date
- **Primary Key** : {(suggestion_id)}
- **Candidate Keys** : {(suggestion_id)}
- **Normal Form** : BCNF
- **Table** :

```
CREATE TABLE Suggestion (
        suggestion_id INT AUTO_INCREMENT,
        message VARCHAR(255),
        date DATE,
        PRIMARY KEY (suggestion_id)) ENGINE = InnoDB;
```

4.22. Baseballer Information

- **Relation Schema:**
  BaseballerInformation (<u>date, park, baseballer</u>, position, substitution_first, substitution_last, game_time(), baseballer_score)
- **Function Dependencies:**
  date, park, baseballer → position, substitution_first, substitution_last, game_time(), baseballer_score
- **Primary Key** : {(date, park, baseballer)}
- **Candidate Keys** : {(date, park, baseballer)}
- **Normal Form** : BCNF
- **Table** :

```
CREATE TABLE BaseballerInformation(
        date DATE,
        park VARCHAR(40),
        baseballer INT,
        position VARCHAR(3),
        substitution_first TIME,
        substitution_last TIME,
        game_time TIME,
        baseballer_score MEDIUMINT,
        PRIMARY KEY (date, park, baseballer),
        FOREIGN KEY (date, park) REFERENCES Competition,
        FOREIGN KEY (baseballer) REFERENCES Baseballer(baseballer_id),
        FOREIGN KEY (position) REFERENCES Baseballer(position))
        ENGINE = InnoDB;
```

4.23. Punishment

- **Relation Schema:**
  Punishment (<u>date, park,</u> baseballer, punishment)
- **Function Dependencies:**
  date, park → baseballer, punishment
- **Primary Key** : {(date, park)}
- **Candidate Keys** : {(date, park)}
- **Normal Form** : BCNF
- **Table** :

```
CREATE TABLE Punishment(
        date DATE,
        park VARCHAR(40),
        baseballer INT,
        punishment VARCHAR(6),
        PRIMARY KEY (date, park, time, baseballer),
        FOREIGN KEY (date, park) REFERENCES Competition,
        FOREIGN KEY (baseballer) REFERENCES Baseballer(baseballer_id))
        ENGINE = InnoDB;
```

4.24. Message

- **Relation Schema** :
  Message(<u>message_id</u>, message_text, date, support_id)
- **Function Dependencies:**
  message_id → message_text, date, support_id
- **Primary Key** : {(message_id)}
- **Candidate Keys** : {(message_id)}
- **Normal Form** : BCNF
- **Table** :

```
CREATE TABLE Message (
        message_id INT AUTO_INCREMENT,
        message_text VARCHAR(255) NOT NULL,
        date DATE,
        support_id INT NOT NULL,
        FOREIGN KEY(support_id) REFERENCES Human(human_id),
        PRIMARY KEY (message_id)) ENGINE = InnoDB;
```

4.25. Suggests

- **Relation Schema:**
  Suggests(sug_id, support_id)
- **Function Dependencies:** No FD
- **Primary Key** : { sug_id, support_id)}
- **Candidate Keys** : {(sug_id, support_id)}
- **Normal Form** : BCNF
- **Table** :

```
CREATE TABLE Suggests (
        sug_id INT NOT NULL,
        support_id INT NOT NULL,
        PRIMARY KEY (sug_id, support_id),
        FOREIGN KEY(sug_id) REFERENCES Suggestion(sug_id),
        FOREIGN KEY(support_id) REFERENCES Human(human_id))
        ENGINE = InnoDB;
```

5. NORMAL FORMS

It appears that all tables have normal forms BCNF.

6. TRIGGERS

6.1. When a person is added to social media, the e-mail must be in the specified format, their gender must be either male or female or OT meaning Other. No other option is accepted by the app.

```
CREATE TRIGGER IF NOT EXISTS control_before_insert_human
  BEFORE INSERT ON human
BEGIN SELECT
    CASE
       WHEN NEW.email NOT LIKE '%_@__%.__%' THEN RAISE (ABORT,'Invalid
email address')
       WHEN (NEW.gender not in ("male",”female", “OT”)) THEN RAISE
(ABORT,'Invalid gender type')
     END;
END;
```

6.2. When field manager is added as inheritance logic, human should also be added.

```
CREATE TRIGGER IF NOT EXISTS add_human_after_field_manager
  AFTER INSERT ON fieldmanager
BEGIN
   INSERT INTO human (huma_id, fullname, e-mail, username, password, nationality)
   VALUES (NEW.field_manager_id, “”, “”, “”, null, “”);
   UPDATE FieldManager SET
   human_id = CASE
        WHEN (SELECT seq FROM sqlite_sequence WHERE name="human")>0 THEN
(SELECT seq FROM sqlite_sequence WHERE name="human")+1
        ELSE 1
   END

   WHERE field_manager_id=NEW.field_manager_id;
END;
```

NOTE : There are many triggers like this. However, only one is shown in the report.

6.3. When a Baseball substitution is made, the statistics of the relevant players are updated. After this update, the player's playing time is calculated according to the entry and exit time.

```
CREATE TRIGGER IF NOT EXISTS baseballer_game AFTER INSERT ON
BaseballerInformation
FOR EACH ROW
new.game_time = new.substitution_last - new. substitution_first
```

6.4. It is the trigger that will work when people are deleted from social media. It is important. Because the person in question could be anyone from a manager to a baseballer.

```
CREATE TRIGGER IF NOT EXISTS delete_human_from_social_databases
  AFTER DELETE ON Human
BEGIN
   Delete from human where human_id=OLD.human_id;
END;
```

6.5. Teams need to update their respective leaderboards for posting a match.

```
CREATE TRIGGER update_team_statistics AFTER INSERT ON Competition
FOR EACH ROW
BEGIN
    IF NOT EXISTS (SELECT * FROM TeamStatistics
            WHERE tournament_id = new.Tournament AND team_id = new.team1)
    INSERT INTO TeamStatistics values(new.Tournament, new.team1, 0, 0, 0, 0, 0, 0);

    IF NOT EXISTS (SELECT * FROM TeamStatistics
            WHERE tournament_id = new.Tournament AND team_id = new.team2)
    INSERT INTO TeamStatistics values(new.Tournament, new.team2, 0, 0, 0, 0, 0, 0);

    UPDATE TeamStatistics
    SET gain = gain + IF(new.team1_score > new.team2_score, 1, 0),
            lost = lost + IF(new.team1_score < new.team2_score, 1, 0),
            scores_for = scores_for + new.team1_score,
            scores_against = goals_against + new.team2_score
    WHERE tournament_id = new.Tournament AND team_id = new.team1;

    UPDATE TeamStatistics
    SET gain = gain + IF(new.team2_score > new.team1_score, 1, 0),
            lost = lost + IF(new.team2_score < new.team1_score, 1, 0),
            scores_for = scores_for + new.team2_score,
            scores_against = scores_against + new.team1_score
    WHERE tournament_id = new.Tournament AND team_id = new.team2;

    UPDATE TeamStatistics
    SET competitions = competitions + 1,
        scores_diff = scores_for - scores_against,
        points = won
    WHERE tournament_id = new.Tournament AND (team_id = new.team1 or team_id =
    new.team2);
END
```

NOTE : The number of triggers may increase with the interface. These are the triggers that are planned to be shown for now.

7. VIEWS

7.1. Transfer offers to Baseballers can be seen by club directors.

```
CREATE VIEW ongoing_transfers
AS SELECT date, baseballer, club_sell, value, club_buy, value, value_offer
FROM ((TransferRequest NATURAL RIGHT OUTER JOIN ManagerChanges)
    JOIN baseballer ON baseballer = baseballer_id)
    JOIN team ON (team_id = club_sell OR team_id = club_buy)
```

7.2. In this view, he again finds the winners of the tournaments. However, if there are teams with the same score in the league-style tournament, it is based on the score differences.

```
CREATE VIEW winners
AS SELECT tournament_id, name, start, end, team_id
FROM Tournament NATURAL JOIN (
    TeamStatistics NATURAL JOIN (
        SELECT tournament_id, scores, MAX(score_diff) AS score_diff
        FROM TeamStatistics NATURAL JOIN (
            SELECT tournaments_id, MAX(scores) AS scores
            FROM TeamStatistics
            GROUP BY tournament_id)
        GROUP BY tournaments_id, scores))
WHERE end_date < curdate()
```

7.3. Supporters can view the tournaments that the teams they support are currently participating in, and the scoreboard.

```
CREATE VIEW now_tournaments
AS SELECT *
FROM tournaments NATURAL JOIN TeamStatistics
WHERE end_date > curdate()
```

7.4. It can be thought of as the information that would appear on the page summarizing a baseballer's career.

```
CREATE VIEW baseballer_career (id, name, cup, year)
AS SELECT baseballer_id, fullname, name, year(end_date)
FROM (agreement JOIN champions) JOIN baseballer on baseballer_id = human_id
WHERE team_id = club_buy AND (end between start AND end)
```

7.5. It allows to display the winner team in the relevant tournament.

```
CREATE VIEW winners
AS SELECT tournament_id, name, start, end, team_id
FROM Tournament NATURAL JOIN (
    TeamStatistics JOIN (
        SELECT tournament_id, MAX(scores) AS max_point
        FROM TeamStatistics
        GROUP BY tournament_id) ON scores = max_point)
WHERE end_date < curdate()
```

NOTE : These views are the ones that are taken for granted. Any changes due to the interface will be mentioned.

8. INTERFACE

Interface work continues. An interface with PHP is being considered.