UNIVERSITY
of
GREENWICH

Student Name: Tran Duc Huy
Student ID: GC60361
University Banner ID: 000962917
Coursework Title: Football Management System.
Course: COMP1639 Database Engineering
Due Date: 21/11/2016

Table of Contents

a) **A brief description of the database including any assumptions made during the design.**

Now a day, the football sound look the best popular with the audition they have just watch it on the TV, and judge it but they do not know with the IT it is hard problem must be face with it. To build the management system logic for the football sport. The developer must acknowledge about football management system.

And now in the role is the IT and responsibility football management system I will build reliable database by develop RDBMS. I hope it can make increase the logic and management for manage system in the plan project I expect divide kinds of three difference management, first the team management, the second is the game management, finally is the information management.

How to design database system?

The first I will approach (Entity Relation Diagram) ERD use the tool to dawn ERD to analysts each kinds of the management and explain the important elements in the management system. In the team management will management: clubs, coaches. In the game management: leagues, scores, matches, players, officials. In the information management is club report, player report. In each kinds of the management have the important attributes.

b) An ERD (Entity Relationship Diagram) that fully describes the database.

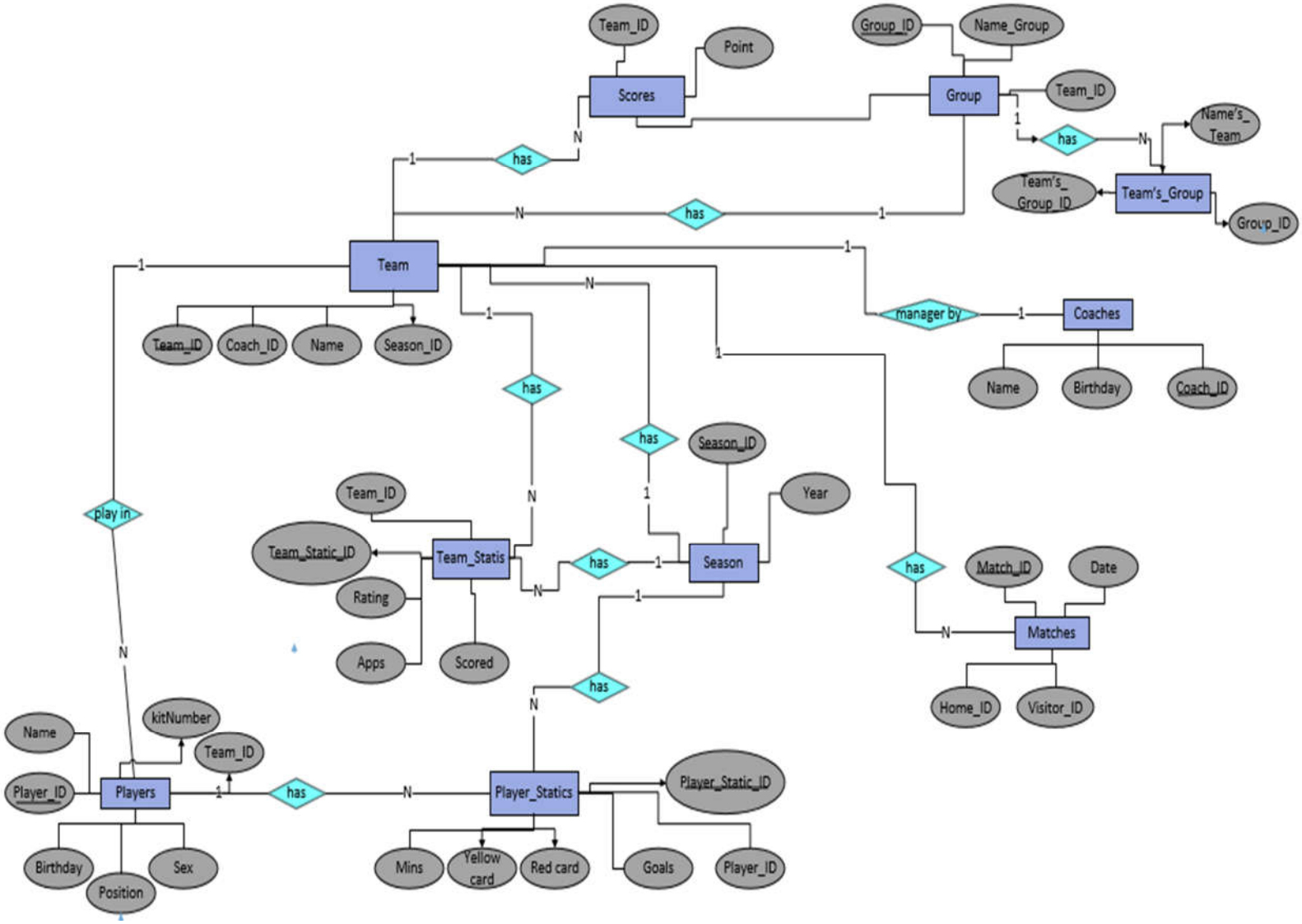

Figure1: Entity Relationship Diagram for Football Management System

- **The ER diagram to build up as above, and then we give describe the details and the meaning of the attributes**

➢ **Player**

- Player_ID: This is primary key, ID of each players to check do not duplicate.
- Name: The name of player
- Birthday: Date of player's birth.
- Position: the position player can activity.
- Sex: the player's gender.
- Team_ID: This is foreign key of Player table
- KitNumber: The player's number was decides on the shirt.

- **Coaches**
  - Coaches_ID: This is primary key, ID of each coaches to check do not duplicate.
  - Name: The name of coaches
  - Birthday: Date of Coach's birth.
- **Team**
  - Team_ID: This is primary key, ID of each Team to check do not duplicate.
  - Name: The name of clubs
  - Coaches_ID: This is foreign key of Team table
  - Season_ID: This is foreign key of Team table
- **Group**
  - Group_ID: This is primary key, ID of each Group to check do not duplicate.
  - Name: The name of group
  - Team_ID: This is foreign key of group table
- **Team's_Group**
  - Team's_Group_ID: This is primary key, ID of each Team's_Group to check do not duplicate.
  - Name: The name of team's_group
- **Matches**
  - Matches_ID: This is primary key, ID of each matches to check do not duplicate.
  - Date: Date of the match active
  - Home_ID: This is foreign key of Matches table
  - Vistor_ID: This is foreign key of Matches table
- **Season**
  - Season_ID: This is primary key, ID of each season to check do not duplicate.
  - Year: year of season.
- **Scores**
  - Point: point of the matches.
  - Team_ID: This is foreign key of Score table
- **Team_Statics**
  - Team_Statics_ID: This is primary key, ID of each Team_Statics_ID to check do not duplicate.
  - Team_ID: This is foreign key of Team_Statics table
  - Rating: the effect of matches
  - Score: season's score
  - Apps: The times join to the game
- **Player_Statics**
  - Player_Repor_ID: This is primary key, ID of each Player_Repor_ID to check do not duplicate.

- Player_ID: This is foreign key of Player_Repor table
- YellowCard: The total yellow card was awarded.
- RedCard: The total red card was awarded.
- Mins: The time player join the matches
- Goals: The total goals player have

c) The relational schema derived from the ERD that is at least in 3NF.

The tables below were achieved 3NF

- Table **Team** (TeamID,CoachesID,Name,SeasonID)
- Table **Team_Statics (**Team_StaticID, Rating, Apps, Score, TeamID**)**
- Table **Group** (GroupID, Name, TeamID)
- Table **Team's_Group** (Team_GroupID, Name, GroupID)
- Table **Season** (SeasonID, Year)
- Table **Player** (PlayerID, Name, Birthday, Position, Sex, KitNumber , TeamID)
- Table **PlayerStatics** (Player_StaticID, Mins, Goals, RedCard, YellowCard, Player_ID)
- Table **Matches**(MatchID, Date, HomeID, VisitorID)
- Table **Scores**(Point, TeamID)

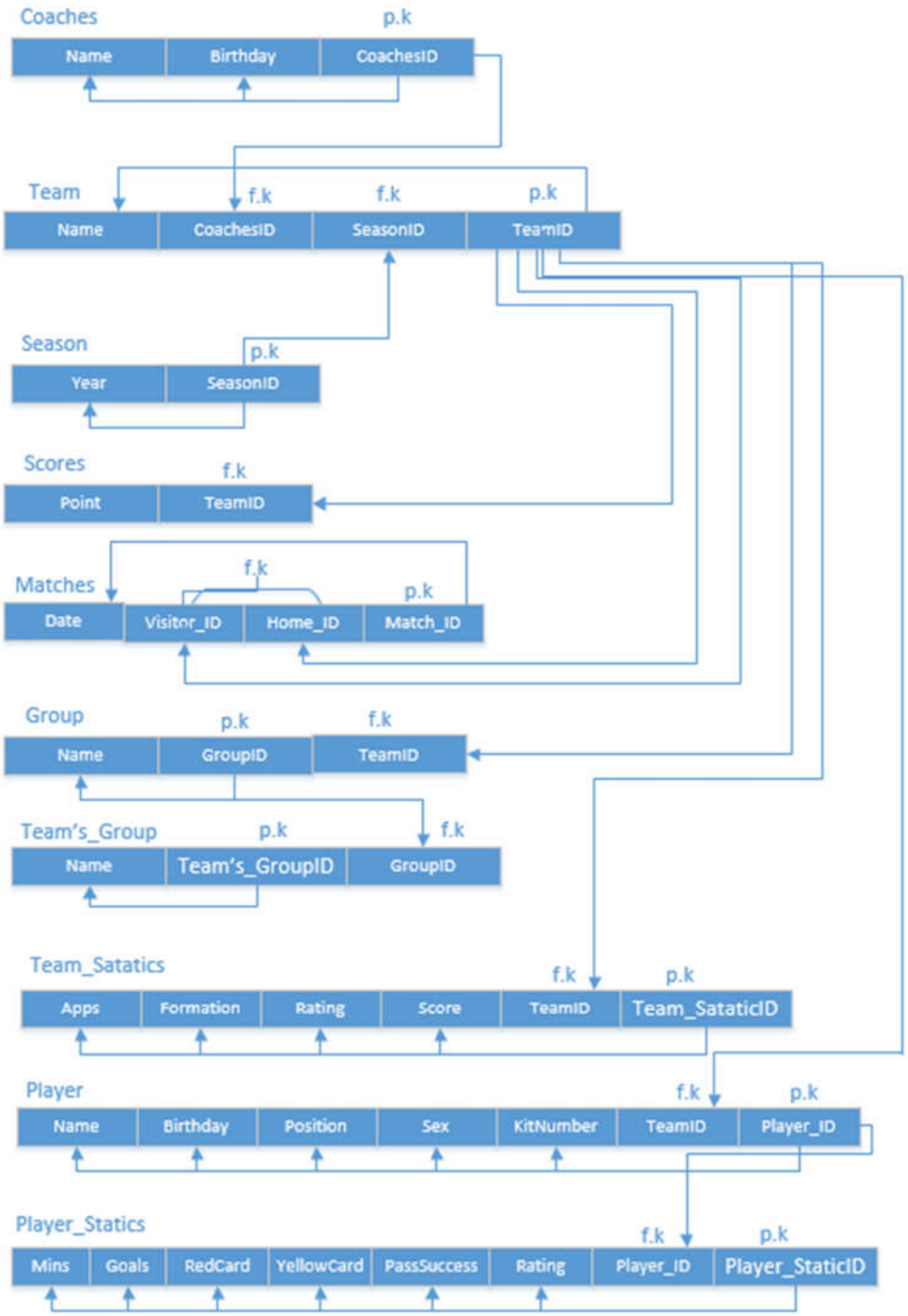

Figure 2: Relational schema

d) The set of database statements used to create the tables, indexes, triggers and views used in your database. You do NOT need to include all the data and insert statements.

- Some statement **Create Table**

```
Create table Coaches(
CoachID int  IDENTITY(1,1) NOT NULL primary key,
Name nvarchar (30) NOT NULL,
Birthday nvarchar (30) NOT NULL
)
Go
```

```
Create table Team(
TeamID int IDENTITY(1,1) NOT NULL primary key,
CoachesID int NOT NULL FOREIGN KEY REFERENCES Coaches(CoachID),
Name nvarchar (30) NOT NULL,
Season int NOT NULL FOREIGN KEY REFERENCES Season(SeasonID)
)
Go
```

- Some Statement **Create Index**

```
Create INDEX MatchIndex on Matches ([Date], HomeID, VistorID)
Create INDEX PlayerIndex on Player (PlayerID,TeamID, Name, Position)
Create INDEX TeamIndex on Team (TeamID, Name)
```

- Some Statement **Create View**

```
Create VIEW [Name of player] as
select PlayerID, Name
From Player
Go
```

```
Create View [Report Player] as
select PlayerID,  YellowCard, RedCard, Goals
From Player_Statics
```

- Some Statement **Create Trigger**

AFTER INSERT Trigger

```
CREATE TRIGGER trgAfterInsert ON [dbo].[Coaches]
FOR INSERT
AS
   declare @coachesID int;
   declare @name varchar(100);
   declare @Birthday varchar(100);
   declare @season int;

   select @coachesID =i.coachID from inserted i;
   select @name=i.Name from inserted i;
   select @Birthday=i.birthday from inserted i;
   set @season=1 ;

   insert into Team
        (teamID,Name,coachesID,season)
   values(@coachesID ,@name,@Birthday,@season);

   PRINT 'AFTER INSERT trigger fired.'
GO
```

AFTER DELETE Trigger

```
CREATE TRIGGER trgAfterInsert ON [dbo].[Coaches]
FOR INSERT
AS
   declare @coachesID int;
   declare @name varchar(100);
   declare @Birthday varchar(100);
   declare @season int;

   select @coachesID =i.coachID from deleted i;
   select @name=i.Name from deleted i;
   select @Birthday=i.birthday from deleted i;
   set @season=1 ;

   insert into Team
        (teamID,Name,coachesID,season)
   values(@coachesID ,@name,@Birthday,@season);

   PRINT 'AFTER INSERT trigger fired.'
GO
```

```
ALTER TABLE Team ENABLE TRIGGER ALL
```

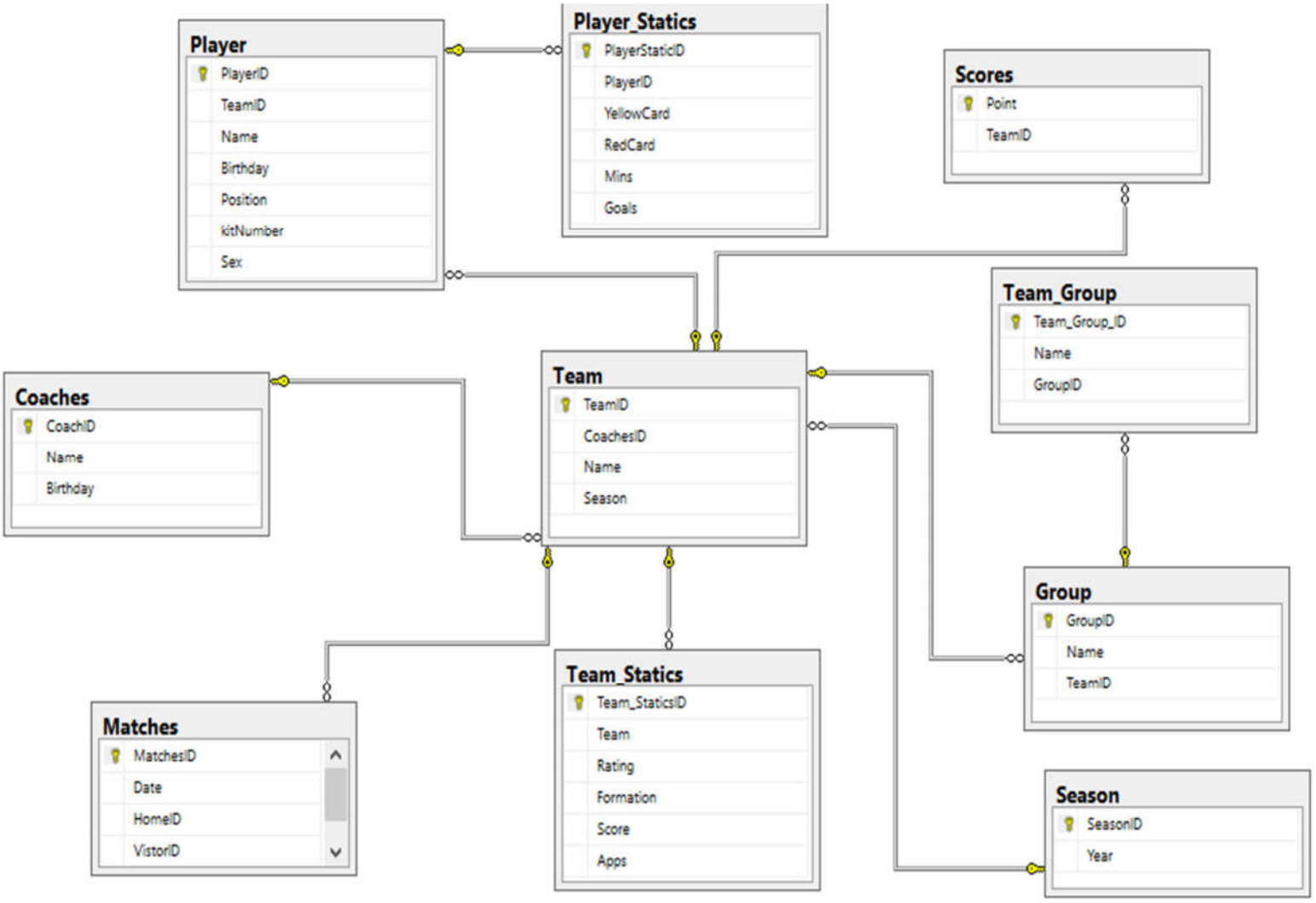

Figure 3: Database diagram

e) 10 queries that demonstrate the usefulness of the database. Also state why and when each query would be used. The following must be demonstrated by at least one of your queries

1. A query that **ORDER BY**

```
Select * from Player_Statics Order by YellowCard ASC, Mins ASC;
```

	PlayerStaticID	PlayerID	YellowCard	RedCard	Mins	Goals
1	4	4	0	1	13p	1-goal
2	2	2	0	0	53p	1-goal
3	1	1	1	0	23p	0-goal
4	3	3	3	2	83p	0-goal

From following the statement Player_Statics table the both of column “YellowCard” and the column “Mins” will ascending.

2. A query that **INNER JOINS**

```
Select Team.Name, Team.TeamID , Scores.Point
from Team
INNER JOIN Scores on Team.TeamID = Scores.TeamID
```

	Name	TeamID	Point
1	team 6	2	1
2	team 5	1	3
3	team 6	2	4
4	team 5	1	5

When use INNER JOIN query in SQL it mean the tables can connection together and we can choose some element make it connect to view or control.

3. A query that **uses aggregate functions**

```
SELECT AVG (GroupID),Name
FROM Team_Group
GROUP BY Name;
```

	(No column name)	Name
1	2	Team A
2	2	Team B
3	1	Team C
4	1	Team D
5	1	Team E
6	1	Team F
7	2	Team Q
8	2	Team W

Return the average value AVG(GroupID) for the each group was decided “GROUP BY” Name and will sum it.

```
SELECT Max (PlayerID),Name
FROM Player
GROUP BY Name;
```

	(No column name)	Name
1	3	Huy
2	2	ngan
3	4	thai
4	1	thang

Return the largest value MAX(PlayerID) for the each group was decided “GROUP BY” Name will output values

4. A query that uses the **GROUP BY** and **HAVING** clauses

GROUP BY clauses

```
select Position
from Player
GROUP BY Position
order by Position desc;
```

	Position
1	vice-leader
2	member
3	leader

Now we can know how many Position of Player in table “Player”

Have Clauses

```
Select Team.Name,count(Scores.Point) as ScoreOfTeam
from( Team INNER JOIN Scores on Scores.TeamID= Team.TeamID)
Group by Name
Having count(Scores.Point)> 1
```

	Name	ScoreOfTeam
1	team 5	2
2	team 6	2

Now we can find the any name of teams have the score >1

5. A query that uses a **sub-query as** a **relation**

```
Select * from Player where TeamID =1;
```

	PlayerID	TeamID	Name	Birthday	Position	kitNumber	Sex
1	3	1	Huy	27/1/1995	member	8	Male
2	4	1	thai	31/3/1993	member	9	Male

In table “Player” we can find and know the players which they in the team.

6. A query that uses a **sub-query** in the **WHERE** clause

```
Select * From Player where PlayerID in ( select PlayerID from Player where
Position = 'a' );
```

	PlayerID	TeamID	Name	Birthday	Position	kitNumber	Sex
1	5	1	A	27/1/1995	a	8	Male
2	6	1	B	31/3/1993	a	9	Male
3	7	1	A	27/1/1995	a	8	Male
4	8	1	B	31/3/1993	a	9	Male

We call all information of one value we get is “Position” it will show all Position of Player is “a”

7. A query stored as a **VIEW**

```
CREATE VIEW [PlayerList] AS
SELECT PlayerID,Name
FROM Player
Select * from [PlayerList]
```

	PlayerID	Name
1	1	thang
2	2	ngan
3	3	Huy
4	4	thai
5	5	A
6	6	B
7	7	A
8	8	B

View all the name, ID of “Player” table

8. A query that uses a **VIEW as** a **relation**

```
CREATE VIEW PlayerTeam AS
SELECT * From Player
WHERE TeamID = 1
```

	PlayerID	TeamID	Name	Birthday	Position	kitNumber	Sex
1	3	1	Huy	27/1/1995	member	8	Male
2	4	1	thai	31/3/1993	member	9	Male
3	5	1	A	27/1/1995	a	8	Male
4	6	1	B	31/3/1993	a	9	Male
5	7	1	A	27/1/1995	a	8	Male
6	8	1	B	31/3/1993	a	9	Male

Now we can view player by team ID -> get all player in teamID =1

9. A query that uses **partial matching** in the **WHERE** clause

```
Select Name, Position, kitNumber From Player where Position ='member'
```

	Name	Position	kitNumber
1	Huy	member	8
2	thai	member	9

Choose the element in “Player” by Position =‘member’

10. A query that uses a **self-JOIN**

```
SELECT a.Name,a.TeamID, b.Name, b.GroupID
from [Group] a join Team_Group b
on (a.GroupID= b.Team_Group_ID);
```

	Name	TeamID	Name	GroupID
1	Gruop A	1	Team Q	2
2	Group B	2	Team W	2
3	Gruop A	1	Team A	2
4	Group B	2	Team B	2

Join the tables together and choose element to view it

Host test query in (FSM database) => http://rextester.com/XVC97880