# 1 ER Diagrams

We want to store sports teams and their players in our database. Draw an ER diagram corresponding to data given below:

- Every Team in our database will have a unique team_name and a stadium where they play their games.
- Each Coach has a name.
- Each Player will have a unique player_id, a name and an average score.
- Our database will contain who Plays_For which team and also the "position" that the player plays in. We also need to store who Captains a team, and who Coaches a team.
- Every Team needs players, and needs exactly one captain.
- Each Player can be on at most one team, but may currently be a free agent and not on any team.
- Each team needs coaches and may have many.
- A Coach is uniquely identified by which team they coach.
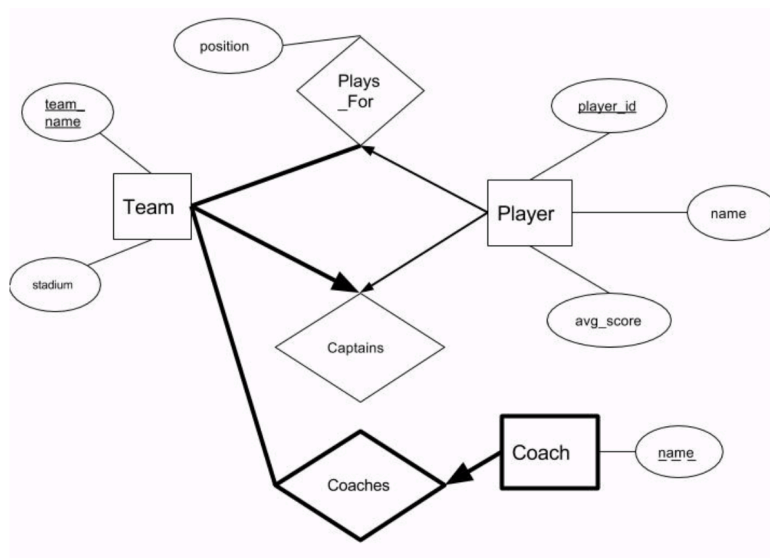
2 Functional Dependencies

1. Consider a relation R(x, y, z) and the list of functional dependencies X -> Y, XY -> YZ, and Y -> X where X = {x}, Y = {y}, and Z = {z}. For each of the following relations, indicate which functional dependencies it might satisfy.

x	y	z
1	2	1
1	3	1
2	3	0

x	y	z
1	3	1
2	3	0

x	y	z
1	3	1

1. None
2. XY -> YZ
3. X -> Y, XY -> YZ
4. X -> Y, XY -> YZ, Y -> X

2. Consider the set F = {A -> B, AB -> AC, BC -> BD, DA -> C} of functional dependencies. Compute the following attribute closures.

   a. A+ ABCD
   b. B+, C+, D+ B, C, D; B, C, and D do not appear alone on the left of any functional dependency, so nothing is in their attribute closures besides themselves.
   c. AB+, AC+, AD+ ABCD; A+ = ABCD, so AX = ABCD for any X.
   d. BC+ BCD
   e. BD+ BD
   f. CD+ CD
   g. BCD+ BCD

3. Consider again the set F of functional dependencies from Question 2. Indicate whether the following sets of attributes are candidate keys, superkeys (but not candidate keys), or neither.

   a. A candidate key
   b. B, C, D neither
   c. AB, AC, AD superkey
   d. BC neither
   e. BD neither
   f. CD neither
   g. BCD neither

3   Normal Forms

1. Decompose R = ABCDEFG into BCNF, given the functional dependency set: F = {AB → CD, C → EF, G → A, G → F, CE → F}.

   AB→CD => decompose ABCDEFG into ABCDEF, ABG
   C→EF => decompose ABCDEF, ABG into ABCD, CEF, ABG
   G→A => decompose ABCD, CEF, ABG into ABCD, CEF, AG, BG
   G→F => decompose ABCD, CEF, AG, BG into ABCD, CEF, AG, BG
   CE → F => no change in relations
   Final relations: ABCD, CEF, AG, BG.

2. Is the above decomposition lossless?

   Yes, because BCNF is always lossless.

3. Does the above decomposition preserve dependencies? Why/why not?

   No, G → F is not represented in the closure of the union of each subrelation's dependencies.