A step-by-step application development

- Develop a database recording information about teams, players, and their fans, including
  - For each team, its name, its players, its team captain (one of its players), and the colors of its uniform.
  - For each players, his/her name.
  - For each fan, his/her name, favorite teams, favorite players, and favorite colors.

A step-by-step application development

- Develop a database recording information about teams, players, and their fans, including
  - For each team, its name, its players, its team captain (one of its players), and the colors of its uniform.
  - For each players, his/her name.
  - For each fan, his/her name, favorite teams, favorite players, and favorite colors.

Step 1 – Entity Types

- Entity types:
  - Teams (consists of tuples, each describes a team)
  - Players (consists of tuples, each describes a player)
  - Fans (consists of tuples, each describes a fan)
  - Colors (consists of tuples, each describes a color)

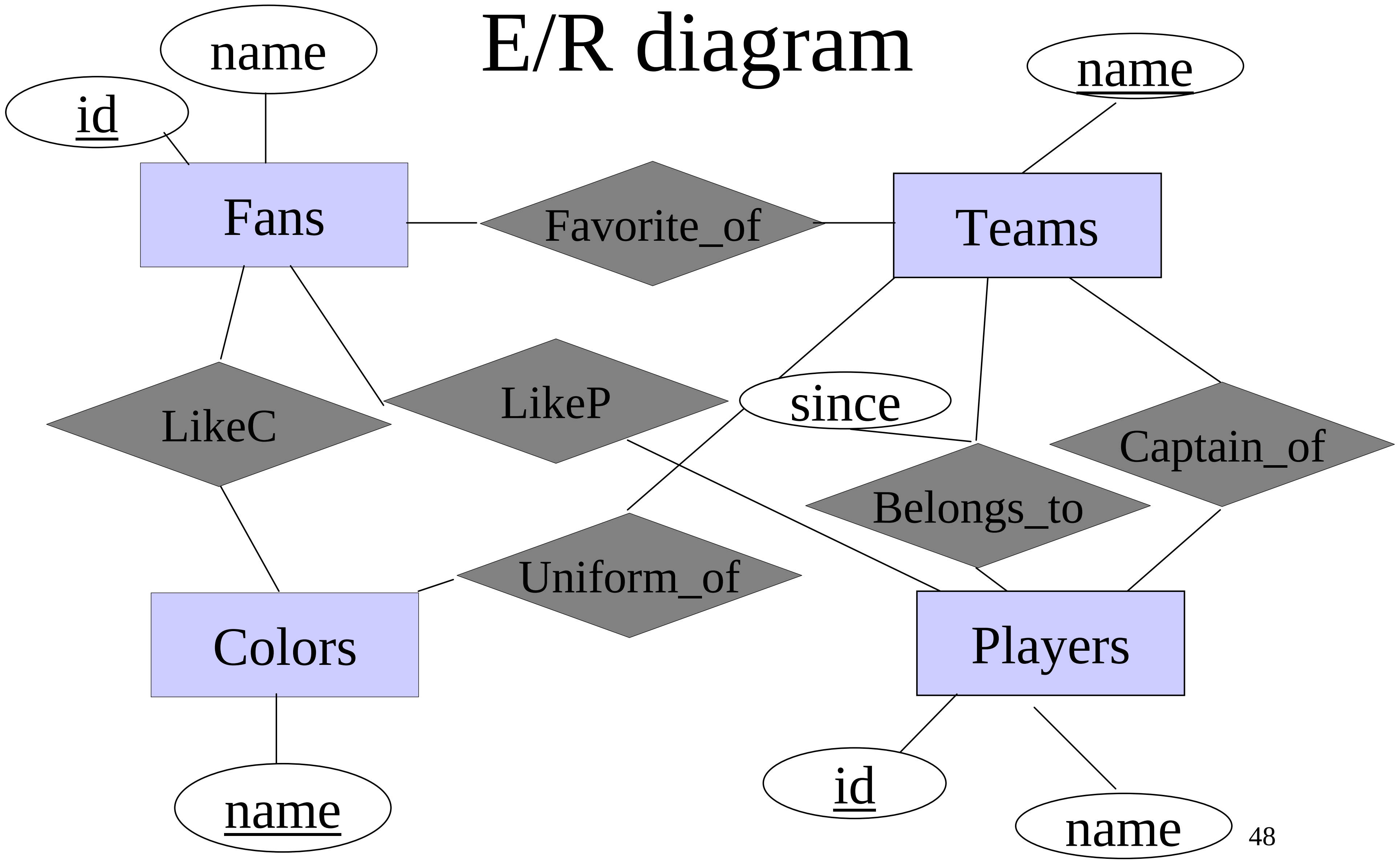
Step 2 – Entity Types and (Their) Attributes

- Entity types:
  - Teams: name   (might be more!)
  - Players: name, id
  - Fans: name, id
  - Colors: name
- Assumptions:
  - Different name for different team
  - No two players with the same id (otherwise?)
  - …

Step 3 – .. and Relationship Types

- Entity types:
  - Teams: name
  - Players: name, id
  - Fans: name, id
  - Colors: name
- Relationship Types:
  - Belongs_to: (Teams, Players, Since) (1:m)
  - Captain_of: (Teams, Players) (1:1)
  - Uniform_of: (Teams, Colors) (1:m)
  - Favorite_of: (Fans, Teams) (n:m)
  - LikeP: (Fans, Players) (n:m)
  - LikeC: (Fans, Colors) (n:m)

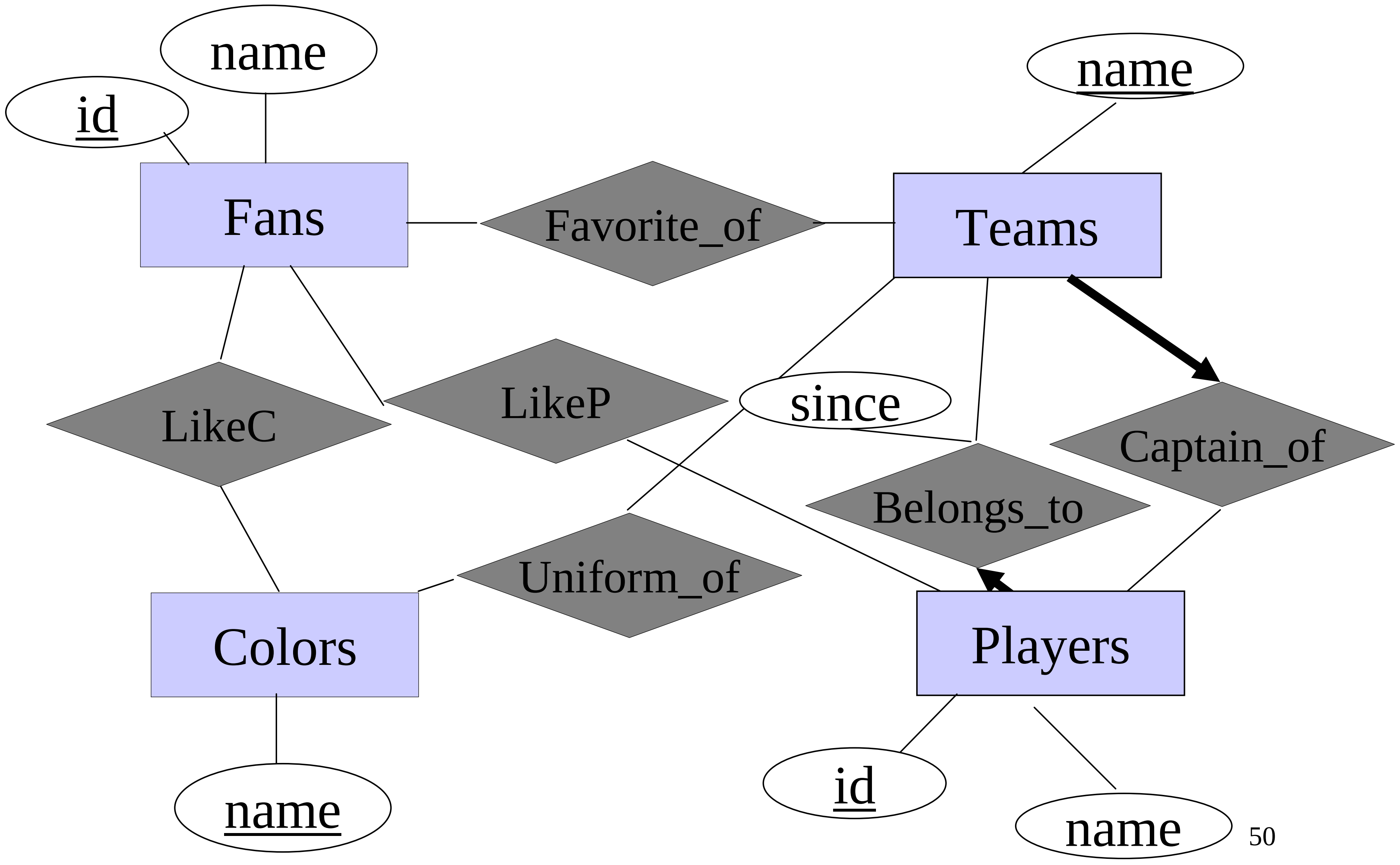
E/R diagram


name
id
Fans
Favorite_of
Teams
name
LikeC
LikeP
since
Captain_of
Belongs_to
Uniform_of
Colors
Players
id
name
name

Adding Constraints

- Each player plays in only one team
  - a participation constraint of tuple in the entity type ‘Players’ with respect to the relationship type ‘Belongs_To’
- Each team has one captain
  - a participation constraint of tuple in the entity type ‘Teams’ with respect to the relationship type ‘Captains’

E/R diagram (with Constraints)



Relations

- Entity types
  - Fans(id, name)
  - Teams(name)
  - Players(id, name)
  - Colors(name)
- Relationship types
  - Belongs_to(teamname, playerid, since) (each player plays for one team)
  - Captain_of(teamname, playerid) (each team has only one captain)
  - Uniform_of(teamname, colorname) (uniform might have >1 color/one color might be used by many teams)
  - Favorite_of(fanid, teamname) (1 fan likes > 1 team/1 team has > 1 fan)
  - LikeP(fanid, playerid) (why?)
  - LikeC(fanid, colorname) (why?)

Constraints

- Belongs_to(teamname, playerid, since)
  - Belongs_to(playerid) **references** Players(id)
- Captain_of(teamname, playerid)
  - Captain_of(teamname) **references** Teams(name)
  - Captain_of(playerid) **references** Players(id)
- Uniform_of(teamname, colorname)
  - Uniform_of(teamname) **references** Teams(name)
  - Uniform_of(colorname) **references** Colors(name)

Constraints

- Favorite_of(fanid, teamname)
  Favorite_of(teamname) **references** Teams(name)
  Favorite_of(fanid) **references** Fans(id)
- LikeP(fanid, playerid)
  LikeP(fanid) **references** Fans(id)
  LikeP(playerid) **references** Players(id)
- LikeC(fanid, colorname)
  LikeC(fanid) **references** Fans(id)
  LikeC(colorname) **references** Colors(name)

Looking back

- Each team has only one captain: *the 'Captain_of' relation contains exactly one tuple for each team*

So, it could be better if we add the captain to the team relation and removes one relation

- Question: if we also want to record the time when the captain being named, something like, *A* is the captain of team *T since 2003*, would it be better for us to keep the 'Captain_of' relation?

SQL (DDL)

```
CREATE TABLE teams(name char(40),
    PRIMARY KEY (name))
CREATE TABLE fans(id integer, name char(40),
    PRIMARY KEY (id))
CREATE TABLE players(id integer, name char(40),
    PRIMARY KEY (id))
CREATE TABLE colors(name char(40),
    PRIMARY KEY (name))
CREATE TABLE belongs_to(pid integer, tname char(40),  since date,
    PRIMARY KEY (pid),
    FOREIGN KEY (pid) REFERENCES players(id),
    FOREIGN KEY (tname) REFERENCES teams(name))
….
```