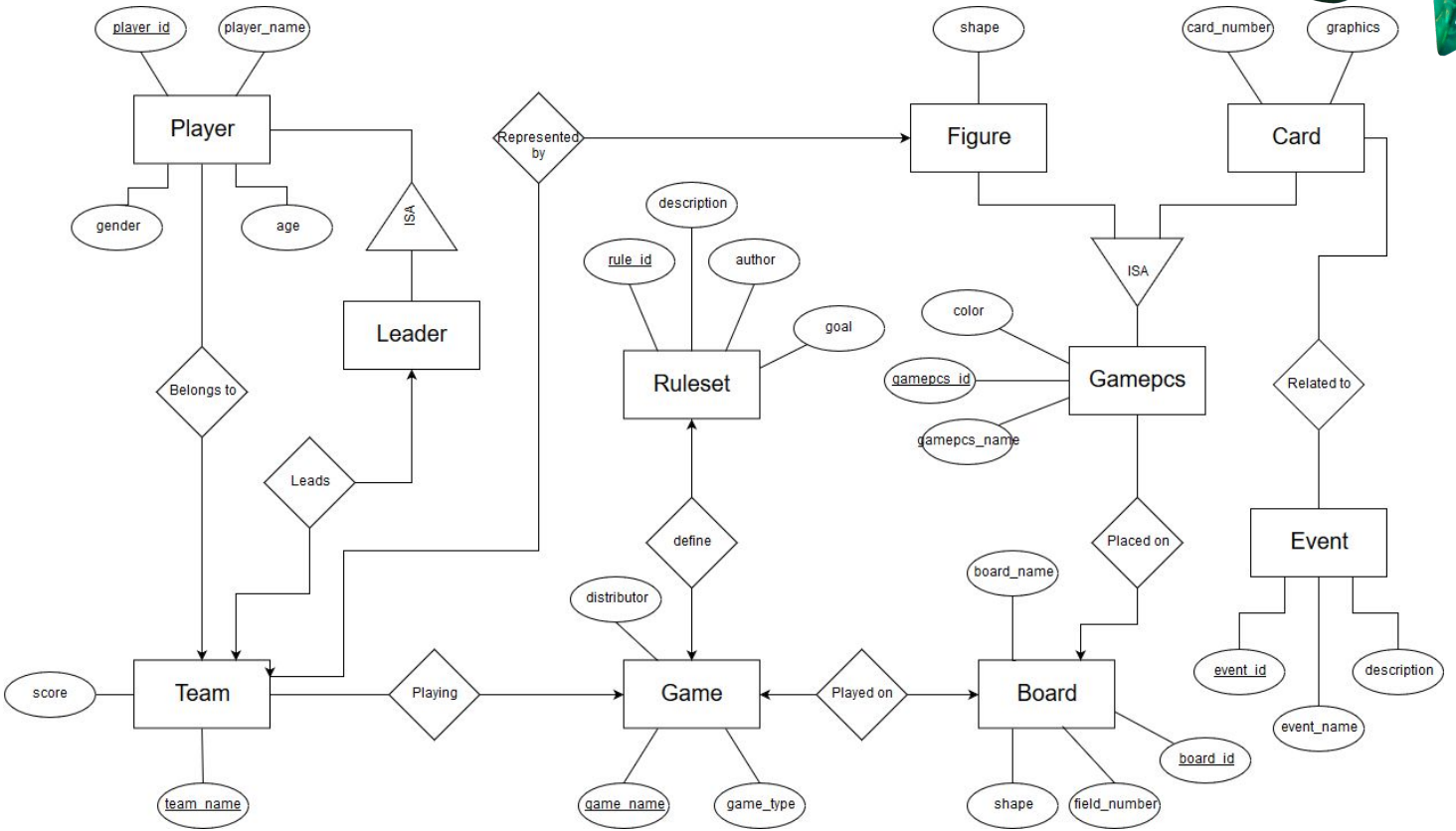
Board game database

Ekart Csaba

Introduction and specification

- My project is demonstrating a database of a simple board game.
- Games are played by teams. Each team is a group of players, and has a leader.
- Board :)
- Special objects like figures, dice, cards, money, tokens etc.
- Figures represent teams, card are related to events
- Ruleset: specifies a goal that a player or the teams has to achive for winning, define and govern the gameplay.

E/R diagram


player_id
player_name
Player
gender
age
ISA
Leader
Belongs to
Leads
Represented by
shape
Figure
card_number
graphics
Card
description
rule_id
author
goal
Ruleset
define
distributor
color
gamepcs_id
gamepcs_name
Gamepcs
ISA
Placed on
Related to
Event
board_name
event_id
description
event_name
score
Team
Playing
Game
Played on
Board
board_id
team_name
game_name
game_type
shape
field_number

Relational model

- **Player**(player_id, name, gender, age, Team.team_name) etc.
- **Related_to**(Card.gamepcs_id, Event.event_id)
- **Gamepcs**(gamepcs_id, gamepcs_name, color, Board.board_id)
    - Gamepcs is an abstract class. The two children (Figure and Card) are in exclusive disjunction, so there is no entity, which can be Figure and Card at the same time.
      Although gamepiece class cannot be instantiated, it makes easier to represent the “Placed on” relationship.

SQL Queries

- What is the name and the description of the event which is related to the yellow cards in the table related to The Duckling Rush?
- Solution*:*

```
SELECT event.event_name, event.description
FROM related_to
INNER JOIN event ON event.event_id = related_to.event_id
INNER JOIN card ON card.gamepcs_id = related_to.gamepcs_id
INNER JOIN board ON board.board_id = card.board_id
INNER JOIN game ON game.game_name = board.game_name
WHERE game.game_name = 'The Duckling Rush' AND card.color = 'yellow';
```

- Result:

```
EVENT_NAME                     DESCRIPTION
------------------------------ --------------------------------------------------
Fly wing                       The teams duck wings has upgraded one level.
```

SQL Queries

- How many girls are older than 10 years in that team, which has The Super Kind Piglet figure and which one is that team?
- Solution*:*

```
SELECT COUNT(player.name), team.team_name
FROM player
INNER JOIN team ON player.team_name = team.team_name
INNER JOIN figure ON team.gamepcs_id = figure.gamepcs_id
WHERE figure.gamepcs_name = 'The Super Kind Piglet' AND player.age > 10 AND player.gender = 'female'
GROUP BY team.team_name;
```

- Result:

```
COUNT(PLAYER.NAME) TEAM_NAME
------------------ ------------------------------
                 2 The Neutral Gophers
```

SQL View

- My example for view lists the game name and the author of its ruleset, which is played by a team with the oldest female leader.

```
CREATE VIEW view_example AS
SELECT game.game_name, ruleset.author
FROM ruleset
INNER JOIN game ON game.rule_id = ruleset.rule_id
INNER JOIN team ON team.game_name = game.game_name
INNER JOIN leader ON leader.leader_id = team.leader_id
WHERE leader.age = (SELECT MAX(AGE) FROM leader WHERE leader.gender = 'female');
```

SQL Trigger

- My trigger is collecting all the changes on the team table into a log file, to see how team scores grow.

```
CREATE TABLE team_log
(
    team_log_time TIMESTAMP PRIMARY KEY,
    team_name VARCHAR2(30),
    team_score NUMBER,
    FOREIGN KEY(team_name) REFERENCES team(team_name)
);
CREATE OR REPLACE TRIGGER team_log_trigger
    AFTER INSERT OR UPDATE
    ON team
    FOR EACH ROW
    BEGIN
        CASE
            WHEN INSERTING THEN INSERT INTO team_log VALUES(systimestamp, :NEW.team_name, :NEW.score);
            WHEN UPDATING THEN INSERT INTO team_log VALUES(systimestamp, :NEW.team_name, :NEW.score);
        END CASE;
    END;
/
```

SQL Trigger

- Result:

	TEAM_LOG_TIME	TEAM_NAME	TEAM_SCORE
1	18-MAY-14 19.56.35,570801000	The Conscious Donkeys	4
2	18-MAY-14 19.56.35,632675000	The Succinct Buffalos	3
3	18-MAY-14 19.56.35,675251000	The Grumpy Elephants	1
4	18-MAY-14 19.56.35,698932000	The Waggish Squirrels	2
5	18-MAY-14 19.56.35,734426000	The Little Shrews	6
6	18-MAY-14 19.56.35,766716000	The Telling Gazelles	4
7	18-MAY-14 19.56.35,812712000	The Fluffy Hedgehoges	10
8	18-MAY-14 19.56.35,840536000	The Kind Panthers	10
9	18-MAY-14 19.56.35,870907000	The Blind Dinos	9
10	18-MAY-14 19.56.35,908320000	The Spotted Whales	9
11	18-MAY-14 19.56.35,929061000	The Crazy Monkeys	7
12	18-MAY-14 19.56.35,970654000	The Sleepy Beavers	3
13	18-MAY-14 19.56.35,999838000	The Strong Camels	1
14	18-MAY-14 19.56.36,028178000	The Sweet Lames	2
15	18-MAY-14 19.56.36,089252000	The Clear Dachshunds	6

Normalization

- Every table has one primary key so all column dependent on the primary key, so BCNF and all other normal forms are fulfilled.
  - **Player**(player_id, name, gender, age, Team.team_name)
  - $F_{player} = \{player\_id \rightarrow name,\ player\_id \rightarrow gender, player\_id \rightarrow age, player\_id \rightarrow Team.team\_name\}$

Normal form	Fulfilled
1NF	OK
2NF	OK
3NF	OK
BCNF	OK

Thank you for your attention!