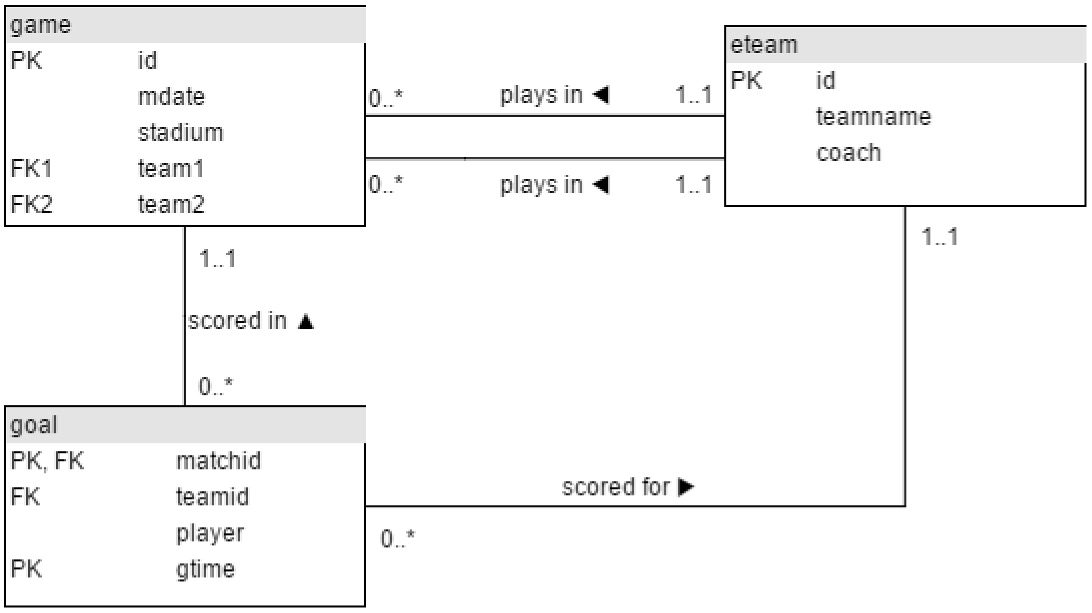
Assignment 1 - SQL Zoo - The JOIN operation

Tags

The JOIN operation



Game Table

id	mdate	stadium	team1	team2
1001	8 June 2012	National Stadium, Warsaw	POL	GRE
1002	8 June 2012	Stadion Miejski (Wroclaw)	RUS	CZE
1003	12 June 2012	Stadion Miejski (Wroclaw)	GRE	CZE

1004	12 June 2012	National Stadium, Warsaw	POL	RUS
...				

Goal Table

matchid	teamid	player	gtime
1001	POL	Robert Lewandowski	17
1001	GRE	Dimitris Salpingidis	51
1002	RUS	Alan Dzagoev	15
1002	RUS	Roman Pavlyuchenko	82
...			

eTeam Table

id	teamname	coach
POL	Poland	Franciszek Smuda
RUS	Russia	Dick Advocaat
CZE	Czech Republic	Michal Bilek
GRE	Greece	Fernando Santos
...		

1.

The first example shows the goal scored by a player with the last name 'Bender'. The `*` says to list all the columns in the table - a shorter way of saying `matchid, teamid, player, gtime`

Modify it to show the *matchid* and *player* name for all goals scored by Germany. To identify German players, check for: `teamid = 'GER'`

```
SELECT matchid, player
FROM goal
```

```
  WHERE teamid = 'GER'
```

2.

From the previous query you can see that Lars Bender's scored a goal in game 1012. Now we want to know what teams were playing in that match.

Notice in the that the column `matchid` in the `goal` table corresponds to the `id` column in the `game` table. We can look up information about game 1012 by finding that row in the **game** table.

Show id, stadium, team1, team2 for just game 1012

```
SELECT id,stadium,team1,team2
  FROM game
where id = 1012;
```

3.

You can combine the two steps into a single query with a `JOIN`.

```
SELECT *
  FROM game JOIN goal ON (id=matchid)
```

The **FROM** clause says to merge data from the goal table with that from the game table. The **ON** says how to figure out which rows in **game** go with which rows in **goal** - the **matchid** from **goal** must match **id** from **game**. (If we wanted to be more clear/specific we could say

`ON (game.id=goal.matchid)`

The code below shows the player (from the goal) and stadium name (from the game table) for every goal scored.

Modify it to show the player, teamid, stadium and mdate for every German goal.

```
SELECT player, teamid, stadium, mdate
  FROM game ga
```

```
JOIN goal go ON ga.id = go.matchid
where teamid = 'GER';
```

4.

Use the same `JOIN` as in the previous question.

Show the team1, team2 and player for every goal scored by a player called Mario `player LIKE 'Mario%'`

```
SELECT team1, team2, player
FROM game ga
JOIN goal go on ga.id = go.matchid
WHERE player LIKE 'Mario%';
```

5.

The table `eteam` gives details of every national team including the coach. You can `JOIN` `goal` to `eteam` using the phrase `goal JOIN eteam on teamid=id`

Show `player`, `teamid`, `coach`, `gtime` for all goals scored in the first 10 minutes `gtime<=10`

```
SELECT player, teamid, coach, gtime
  FROM goal g
JOIN eteam t on g.teamid = t.id
 WHERE gtime <= 10
```

6.

To `JOIN` `game` with `eteam` you could use either

`game JOIN eteam ON (team1=eteam.id)` or `game JOIN eteam ON (team2=eteam.id)`

Notice that because `id` is a column name in both `game` and `eteam` you must specify `eteam.id` instead of just `id`

List the dates of the matches and the name of the team in which 'Fernando Santos' was the team1 coach.

```
SELECT mdate, teamname
FROM game g
JOIN eteam t ON g.team1 = t.id
where coach = 'Fernando Santos';
```

7.

List the player for every goal scored in a game where the stadium was 'National Stadium, Warsaw'

```
SELECT player
FROM game g
JOIN goal go ON g.id = go.matchid
where stadium = 'National Stadium, Warsaw';
```

8.

The example query shows all goals scored in the Germany-Greece quarterfinal.

Instead show the name of all players who scored a goal against Germany.

Select goals scored only by non-German players in matches where GER was the id of either **team1** or **team2**.

You can use `teamid!='GER'` to prevent listing German players.

You can use `DISTINCT` to stop players being listed twice.

```
SELECT distinct player
  FROM game ga JOIN goal go ON go.matchid = ga.id
    WHERE (team1='GER' or team2 ='GER')
and go.teamid <> 'GER';
```

9.

Show teamname and the total number of goals scored.

COUNT and GROUP BY

```
SELECT teamname, count(*)
  FROM eteam t JOIN goal g ON t.id=g.teamid
 group BY teamname;
```

10.

Show the stadium and the number of goals scored in each stadium.

```
select stadium, count(*)
from game g
join goal go on g.id = go.matchid
group by stadium;
```

11.

For every match involving 'POL', show the matchid, date and the number of goals scored.

```
SELECT matchid,mdate, count(*)
  FROM game JOIN goal ON matchid = id
 WHERE (team1 = 'POL' OR team2 = 'POL')
group by matchid, mdate;
```

12.

For every match where 'GER' scored, show matchid, match date and the number of goals scored by 'GER'

```
SELECT matchid, mdate, count(*)
  FROM game JOIN goal ON matchid = id
 WHERE (team1 = 'GER' OR team2 = 'GER') and teamid = 'GER'
group by matchid, mdate;
```

13.

List every match with the goals scored by each team as shown. This will use "CASE WHEN" which has not been explained in any previous exercises.

mdate	team1	score1	team2	score2
1 July 2012	ESP	4	ITA	0
10 June 2012	ESP	1	ITA	1
10 June 2012	IRL	1	CRO	3
...				

Notice in the query given every goal is listed. If it was a team1 goal then a 1 appears in score1, otherwise there is a 0. You could SUM this column to get a count of the goals scored by team1. **Sort your result by mdate, matchid, team1 and team2.**

```
SELECT mdate,
  team1,
  SUM(CASE WHEN teamid=team1 THEN 1 ELSE 0 END) as score1,
team2,
SUM(case when teamid=team2 then 1 else 0 end) as score2
  FROM game LEFT JOIN goal ON matchid = id
group by mdate, matchid
order by mdate, matchid, team1, team2;
```