Nicholas Capriotti

CS317 - Van Hilst

Assignment 3B: Data and Table Design

This assignment is more practice with sub queries and table design. The assignment will require us to make and populate several tables, and perform operations on them using subqueries to get various information. The assignment will require us to also delete the tables when we are done using them. We will create 3 separate files, one for creating/populating, one for the queries, and one for the deletion of the tables.

The first step is to create our 4 tables, we start with the teams table, which will contain a team name and a coach, with the primary key being the team. The primary key assignment to 'team' means that by knowing the team, you will be able to know the associated coach. We also add the NOT NULL condition to the tema variable to make sure that there is no instance where a player or coach has no team. Using NOT NULL with the primary key is a good practice. We will then populate the table with the INSERT INTO command with the values for the team and coach names we choose.

```
CREATE TABLE teams (
    team VARCHAR(32) NOT NULL,
    coach VARCHAR(32),
    PRIMARY KEY(team)
);
INSERT INTO teams VALUES
    ('Cardinals','Newcomb'),
    ('Cowboys','Stowers'),
    ('Rampage','Mussleman'),
    ('Ropers','Capriotti'),
    ('Packers','LaFleaur'),
    ('Bears','Nagy');
```

The next table we want to create is a schedule table, this table will contain a game_id to keep track of which 2 teams are playing each other (whoever shares a game_id will play each other). The table also has the same team variable as the prior table, this will allow us to do join operations on the table later in the assignment. There is also a bool character that is true if the team is hosting (bringing snacks) or a 0 if they are not hosting. There is a variable to keep score as well. The primary keys are the game_id, and host but we have a new item, the foreign key, which references the team variable of the team table. Foreign keys refer to the primary keys of other tables for relational operations. We do the 'ON UPDATE CASCADE' so that inserting new data in the table won't affect our queries or other data already in the table.

```
CREATE TABLE schedule (
    game_id int NOT NULL AUTO_INCREMENT,
    team VARCHAR(32),
    host bool NOT NULL,
    score int,
    PRIMARY KEY (game_id, host)
    -- FOREIGN KEY(team) REFERENCES teams(team) --
    -- ON UPDATE CASCADE --
);
INSERT INTO schedule VALUES
    ('1','Cardinals','1','4'),
    ('1','Bears','0','5'),
    ('2','Cowboys','1','2'),
    ('2','Packers','0','3'),
```

The next table we create will be our player table, containing a row for each player and their respective team, this means that the primary key needs to be the player, and the foreign key can reference the team column of the teams table. We will then populate the table with player names and assign them to their team using the INSERT INTO command. It is worth noting that the assignment would have preferred separate first and last names for the players, this was an oversight on my end and it is something that could be implemented moving forward.

```
CREATE TABLE players (
    player VARCHAR(32) NOT NULL,
    team VARCHAR(32),
    PRIMARY KEY(player)
    -- FOREIGN KEY(team) REFERENCES teams(team) --
    -- ON UPDATE CASCADE --
);
INSERT INTO players VALUES
    ('Gage C.','Cardinals'),
    ('Baylor R.','Cardinals'),
```

The last table we want to create is the games table, which contains all of the information pertaining to specific games, including the game_id, team, date and location. While this table contains only 1 entry per game, it should be noted that in the table produced by the queries, we will receive 2 rows per game, both with the same game_id, but each one will pertain to one specific team. For example, the game with game_id = 1 will have one entry in the games table, but there are 2 teams associated with that game, one for each side.

```
CREATE TABLE games (
    game_id int NOT NULL AUTO_INCREMENT,
    location VARCHAR(32),
    date DATETIME,
    PRIMARY KEY (game_id)
);
INSERT INTO games VALUES
    ('1', 'Field 1', '2022-10-01 09:00:00'),
    ('2', 'Field 2', '2022-10-01 09:00:00'),
```

Our next step is to create a file that will delete all of the tables and associated data. There is a condition: IF EXIST, that checks our database for tables with the same name, and if those tables exist, they are deleted and the data is lost. We want to run this program before creating tables to ensure there aren't any duplicate names, data, or tables.

```
DROP TABLE IF EXISTS games;
DROP TABLE IF EXISTS players;
DROP TABLE IF EXISTS schedule;
DROP TABLE IF EXISTS teams;
```

For our first query, we want to print each player with their team name, and associated coach. We will use the player table as the primary table, and join it with the teams table with the reference column being the team. This gives us access to the coach for each team so we can put them together. Below is the query with the associated table that was printed from it.

```
SELECT player, teams.team, coach
FROM players
JOIN teams on teams.team = players.team;
```

player	team	coach
Brett F.	Bears	Nagy
Joe N.	Bears	Nagy
Jordan L.	Bears	Nagy
Master C.	Bears	Nagy
Throckmorton S.	Bears	Nagy
Baylor R.	Cardinals	Newcomb
Gage C.	Cardinals	Newcomb
Gunner C.	Cardinals	Newcomb
Mack J.	Cardinals	Newcomb
Tyler T.	Cardinals	Newcomb
Aaron R.	Cowboys	Stowers
Donald T.	Cowboys	Stowers
Juice W.	Cowboys	Stowers
Kanye W.	Cowboys	Stowers
Michael J.	Cowboys	Stowers
Chris P.	Packers	LaFleaur
Clark N.	Packers	LaFleaur
Josiah K.	Packers	LaFleaur
Kirby N.	Packers	LaFleaur
Robert D.	Packers	LaFleaur
Ian C.	Rampage	Mussleman
Joe B.	Rampage	Mussleman
Josh D.	Rampage	Mussleman
Mich J.	Rampage	Mussleman
Pablo C.	Rampage	Mussleman
DeShaun W.	Ropers	Capriotti
Jack L.	Ropers	Capriotti
Jonsey L.	Ropers	Capriotti
Josh Z.	Ropers	Capriotti
Kyler M.	Ropers	Capriotti

The next question asks us to produce a query that gives us a table with a player name, all of the dates for their upcoming games, and a column that tells us if that player's team is the host or not. We start by selecting the 3 columns we want to produce, we then select which table is our primary table. We also then join the player table to the schedule table throughout the ‘team’ column. We can do another join between the schedule and games table because we were given access to the schedule table with the previous join. We can then use the WHERE clause to only pull data for a certain player, in this case, we have the schedule for ‘Aaron R.’, the resulting table is shown below.

```
SELECT player, date, host
FROM players
JOIN schedule ON schedule.team = players.team
JOIN games ON schedule.game_id = games.game_id
WHERE player = "Aaron R.";
```

player	date	host
Aaron R.	2022-10-01 09:00:00	1
Aaron R.	2022-10-08 09:00:00	0
Aaron R.	2022-10-15 09:00:00	0
Aaron R.	2022-10-22 09:00:00	1
Aaron R.	2022-10-29 09:00:00	0

Our next table wants us to show a lot of the same data, but with the addition of telling us who the opponent is, and the coach for the opponent. We can start by using the prior query and using it to create a subquery. This means that our first subquery gives us the player, date, host, game_id, and team. This is done by joining the players table to the schedule table, and the schedule table to the games table. We can mark this table as t1, meaning this table contains a lot of the information we want, but not all. So we will join this created table with another created table that we make by combining the schedule table with the teams table. So we have 2 tables filled with data from other tables, and we can now pick what specific data we want to pull, this is done by the first select statement. We include the WHERE clause at the end to prevent the same team showing up multiple times for the same player. When using subqueries, it's important to note which table you are pulling each column from, that's why we have instances of ‘t1.team’ and ‘t2.team’, each team is referring to a separate table and we don't want to confuse the system . The resulting table is shown below.

```
SELECT player, date, t1.host, t2.team, coach
FROM
(SELECT player, date, host, schedule.game_id, schedule.team
FROM players
JOIN schedule ON schedule.team = players.team
JOIN games ON schedule.game_id = games.game_id
WHERE player = "Aaron R.") as t1
JOIN
(SELECT schedule.team, coach, schedule.game_id
FROM schedule
JOIN teams ON schedule.team = teams.team) as t2
on t2.game_id = t1.game_id
WHERE t1.team != t2.team;
```

player	date	host	team	coach
Aaron R.	2022-10-01 09:00:00	1	Packers	LaFleaur
Aaron R.	2022-10-08 09:00:00	0	Rampage	Mussleman
Aaron R.	2022-10-15 09:00:00	0	Bears	Nagy
Aaron R.	2022-10-22 09:00:00	1	Ropers	Capriotti
Aaron R.	2022-10-29 09:00:00	0	Cardinals	Newcomb

This was not a difficult assignment once I got the chance to sit down with the professor and hear a logical explanation to each step. I had some minor issues with syntax errors, but the concepts learned were not hard to grasp (once I actually paid attention). I think that MySQL is a very powerful tool and I'm starting to understand the real world applications for this program.