CS27020 Assignment - Karl Franks (kpf)

Unnormalised Data Structures

To create the unnormalised data structures I replicated the fields of the two example records, and reworded some of the names of the fields for better clarity. For example, the "name" field to store the players of a team I renamed to "member_name" to clarify it is for a member's name and not the name of anything else, e.g. the team itself.

Team Details

```
team_name
sport
captain_number
captain_name
captain_phone
member_number    )
member_name      ) - these two repeat in groups
training_day     )
training_time    ) - these two are also repeated, in their own groups
```

I have used the team name as the primary key, as it is stated in the introduction of the assignment "each team is for one and only one game or sport" implying team names are unique. I have also given more specific field names where applicable (e.g. "member_number" instead of "number").

The following are the functional dependencies I have identified:

team_name -> sport, training_day, training_time, captain_number, captain_name, captain_phone

The details about the team's sport, when the team trains and details about the captain are dependant on that specific team.

captain_number -> captain_name, captain_phone

The member number of the captain is unique, whereas there may be two players with the same name. As such, captain name and captain phone number are deemed to be dependant on the captain number.

member_number -> member_name

Similarly, a normal player's name may not be unique so is dependant on the number to identify individuals.

Member Details
<u>member_number</u>
member_name
member_phone
sport)
team) - both of these are repeated

As a sports club member number should be unique to each person, and details such as a member's name are not I have assigned the member number as the primary key of this structure.

I have identified the following functional dependencies:

member_number -> member_name, member_phone

The details about the member's name and phone number are guaranteed to be unique, e.g. you may have two members with the same name, or the albeit less likely scenario in today's world that two members have the same phone number listed, perhaps due to not owning a mobile phone and listing their home phone number.

sport -> team

A member may participate in a sport without being part of a team, but if they are part of a team they must be listed as taking part in that sport.

First Normal Form

To bring the unnormalised structures up to first normal form, I had to remove the repeating fields.

Team Details
team_name
sport
captain_number
captain_name
captain_phone

Training
team_name
training_day
training_time

Team Players
team_name
member_number
member_name

Member Details
member_number
member_name
member_phone

Sports Participation
member_number
sport

Team Participation
member_number
team_name

Team Details - I have removed the repeating attributes into two separate tables to store the training details and the details of the players of a specific team.

Training - In this table I have moved the training day and training time fields from the team details table to their own table, and created a composite key of all three fields to ensure there has to be unique combinations of team, day and time.

Team Players - Similar to training, I have moved the member details out of the team details table and given them their own table. I have specified a composite key of the team name, and the member number. As the member name is dependant on the member number, it is not a candidate to be included as part of the key.

Member Details - I have removed the sport and team details, to store only details specific to a member.

Sports Participation - Since a member can participate in multiple sports, I have chosen to create a separate table to store this relationship. This does not store team details however, because as previously state a member can participate in a sport without being on a team. Both fields are included as a composite key, so as to ensure uniqueness.

Team Participation - I have decided to store the member's teams separately due to previously stated reasons surrounding the relationship between teams and sports. I have decided not to store the sport here though, as a team is only for one specific sport so I can simply query the team's details to find its sport as opposed to storing this data twice. As with the sport participation table, both fields form a composite key to ensure uniqueness of data.

Team Participation and Team Players are candidates for being merged or one table deleted, as they have the same key.

Second Normal Form

To bring my structures to second normal form I removed any partial dependencies.

Team Details
team_name
sport
captain_number
captain_name
captain_phone

Training
team_name
training_day
training_time

Team Players
team_name
member_number

Member Details
member_number
member_name
member_phone

Sports Participation
member_number
sport

I have deleted the team participation table, as it and team players stored the same data. I have also removed the member name field from team players, as it is only dependant on the member number.

Third Normal Form

To bring my structures to third normal form, I removed any transitive dependencies.

Team Details
team_name
sport
captain_number

Training
team_name
training_day
training_time

Team Players
team_name
member_number

Team Captain
team_name
captain_number
captain_name
captain_phone

Member Details
member_number
member_name
member_phone

Sports Participation
member_number
sport

To initially achieve third normal form, I have created a new table to store the team captain's details, as the captain name and phone number are dependant on the captain number which is in turn dependant on the team name.

However, I then decided to delete the team captain field altogether, as the captain's details are stored in the member details table as well, so there is no point storing this data twice and to find a captain's phone number for example it would be easier to simply query the member details table.

As such, my final data structures look like this:

Team Details
team_name
sport
captain_number

Training
team_name
training_day
training_time

Team Players
team_name
member_number

Member Details
member_number
member_name
member_phone

Sports Participation
member_number
sport

PostgreSQL Implementation

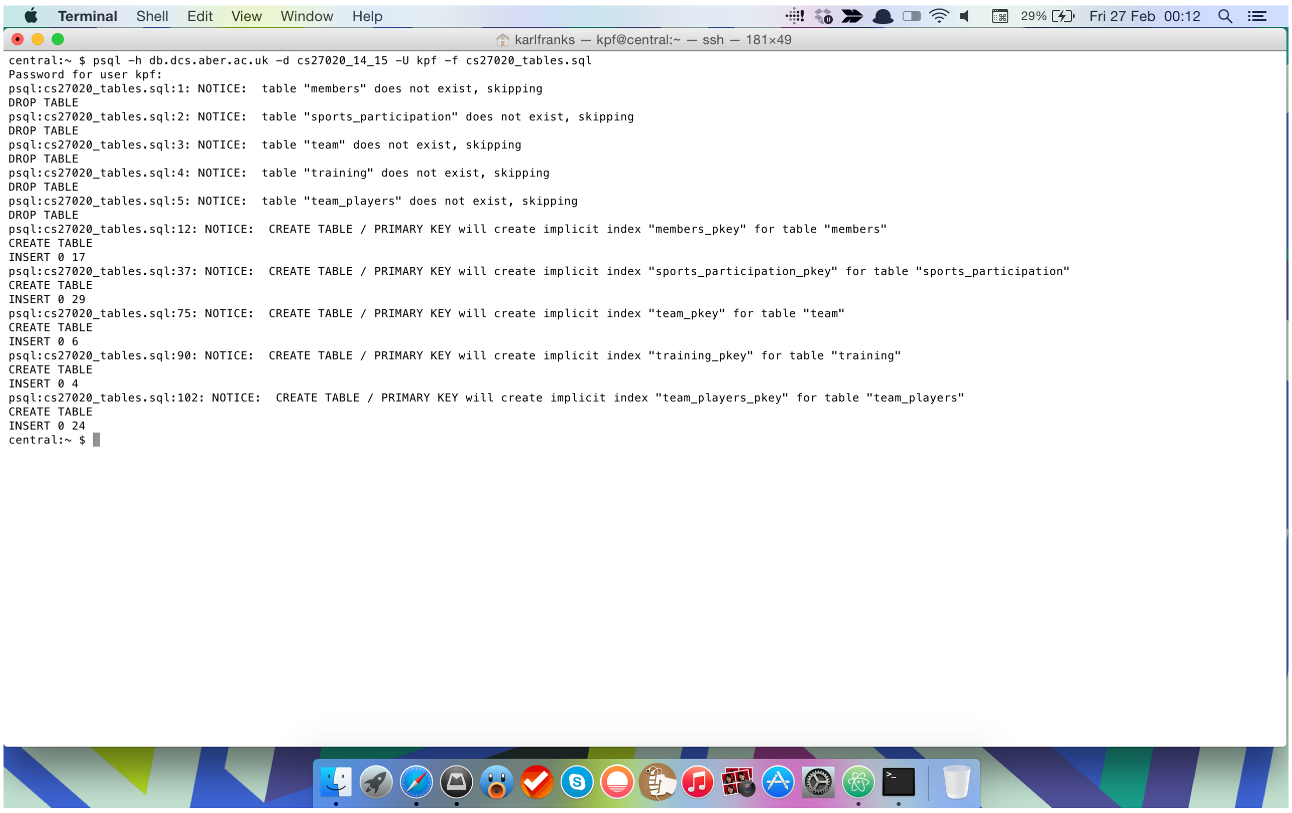
SQL

To implement this structure, I wrote a series of “create table” and “insert into” statements to store in an sql file and run this file, so making any changes was quicker and easier than re-typing every statement into the command line.

This can be found in an appendix at the end of this document.

Running The SQL

As this screenshot shows, when running my “create table” and “insert into” queries was executed with no errors:



```
central:~ $ psql -h db.dcs.aber.ac.uk -d cs27020_14_15 -U kpf -f cs27020_tables.sql
Password for user kpf:
psql:cs27020_tables.sql:1: NOTICE:  table "members" does not exist, skipping
DROP TABLE
psql:cs27020_tables.sql:2: NOTICE:  table "sports_participation" does not exist, skipping
DROP TABLE
psql:cs27020_tables.sql:3: NOTICE:  table "team" does not exist, skipping
DROP TABLE
psql:cs27020_tables.sql:4: NOTICE:  table "training" does not exist, skipping
DROP TABLE
psql:cs27020_tables.sql:5: NOTICE:  table "team_players" does not exist, skipping
DROP TABLE
psql:cs27020_tables.sql:12: NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "members_pkey" for table "members"
CREATE TABLE
INSERT 0 17
psql:cs27020_tables.sql:37: NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "sports_participation_pkey" for table "sports_participation"
CREATE TABLE
INSERT 0 29
psql:cs27020_tables.sql:75: NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "team_pkey" for table "team"
CREATE TABLE
INSERT 0 6
psql:cs27020_tables.sql:90: NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "training_pkey" for table "training"
CREATE TABLE
INSERT 0 4
psql:cs27020_tables.sql:102: NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "team_players_pkey" for table "team_players"
CREATE TABLE
INSERT 0 24
central:~ $
```

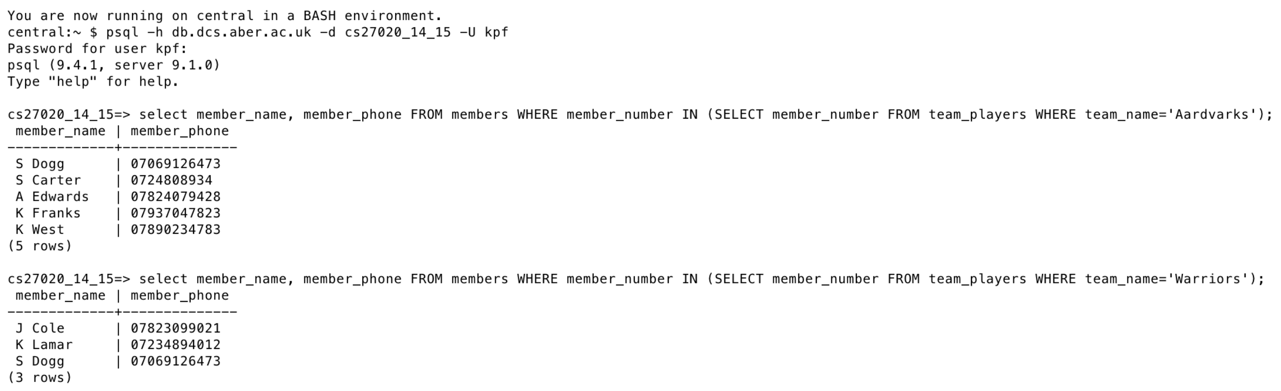
Querying The Database

The assignment stated to created queries to return the members names and phone numbers for a given team, players of a given sport and sports that have no associated team.

Member Query - **SELECT member_name, member_phone FROM members WHERE member_number IN (SELECT member_number FROM team_players WHERE team_name=‘<team name>’);**

This query returns the members names and phone numbers of a team, where the team name is placed between the quotation marks. This is a very simple “SELECT … FROM… WHERE” query, utilising the IN operator to return members whose member number is in a list of selected member numbers associated with a specific team.

An example of this running:

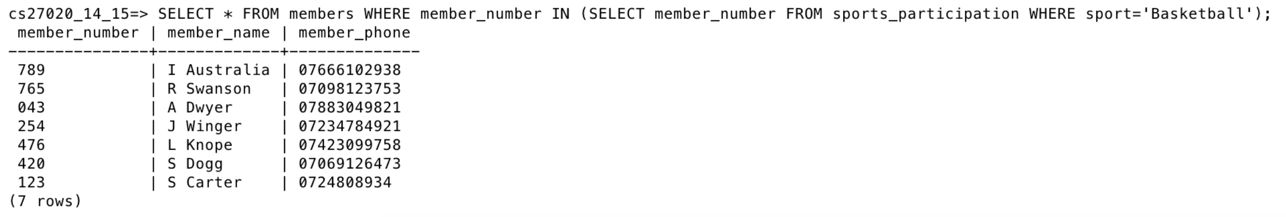


```
You are now running on central in a BASH environment.
central:~ $ psql -h db.dcs.aber.ac.uk -d cs27020_14_15 -U kpf
Password for user kpf:
psql (9.4.1, server 9.1.0)
Type "help" for help.

cs27020_14_15=> select member_name, member_phone FROM members WHERE member_number IN (SELECT member_number FROM team_players WHERE team_name='Aardvarks');
 member_name | member_phone
-------------+--------------
 S Dogg      | 07069126473
 S Carter    | 0724808934
 A Edwards   | 07824079428
 K Franks    | 07937047823
 K West      | 07890234783
(5 rows)

cs27020_14_15=> select member_name, member_phone FROM members WHERE member_number IN (SELECT member_number FROM team_players WHERE team_name='Warriors');
 member_name | member_phone
-------------+--------------
 J Cole      | 07823099021
 K Lamar     | 07234894012
 S Dogg      | 07069126473
(3 rows)
```

Sports Players Query - **SELECT * FROM members WHERE member_number IN (SELECT member_number FROM sports_participation WHERE sport=‘<sport>’);**

Similar to the member query, this will return all the member details of players of a sport, that is placed between the quotation marks. This is a very simple “SELECT … FROM… WHERE” query using the IN operator to return member details of members from a list of member numbers of people associated with a given sport.

An example:

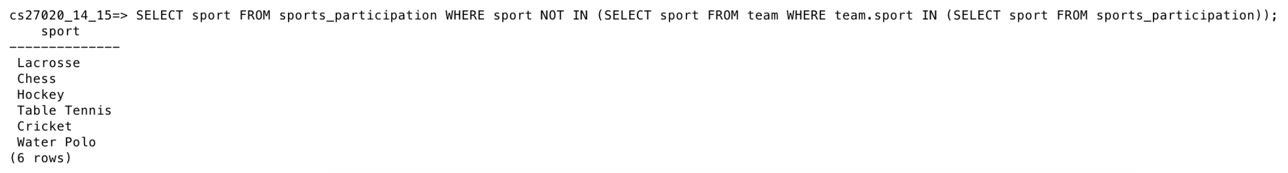


```
cs27020_14_15=> SELECT * FROM members WHERE member_number IN (SELECT member_number FROM sports_participation WHERE sport='Basketball');
 member_number | member_name | member_phone
---------------+-------------+--------------
 789           | I Australia | 07666102938
 765           | R Swanson   | 07098123753
 043           | A Dwyer     | 07883049821
 254           | J Winger    | 07234784921
 476           | L Knope     | 07423099758
 420           | S Dogg      | 07069126473
 123           | S Carter    | 0724808934
(7 rows)
```

Sports With No Teams Query - **SELECT sport FROM sports_participation WHERE sport NOT IN (SELECT sport FROM team WHERE team.sport IN (SELECT sport FROM sports_participation));**

This query is technically three queries nested inside of each other. First I select a list of all the sports people participate in. I then select a list of sports that have an associated team, by using the IN operator again. Lastly, I then return the sports listed in sports participation not in the second query's list by reversing the IN operator using the NOT operator.

An example:



```
cs27020_14_15=> SELECT sport FROM sports_participation WHERE sport NOT IN (SELECT sport FROM team WHERE team.sport IN (SELECT sport FROM sports_participation));
    sport
--------------
 Lacrosse
 Chess
 Hockey
 Table Tennis
 Cricket
 Water Polo
(6 rows)
```

Appendix - SQL

I have written the queries in this order, so as not to attempt to create or populate a table that has a foreign key reference to a table not yet created. For example, if I attempted to create the Training table before the Team table, it would have no team_name field in the Team table to create a foreign key reference.

I also populated it with enough data so as to return a few values on each query, so as to more accurately recreate what the data may look like if this database was implemented for real.

As for data types, I have used text for all fields except training_time which uses the time format. I decided to use text for fields storing numbers so as to ensure the numbers do not get shortened. Eg, phone numbers would have their starting 0 removed. I elected to use text for the training_day field as using a date specifies an absolute date, but sports teams train regularly so it made more sense to store the day of the week as text. I have since learned there are enumerated types in SQL, however I was not aware of this until I saw a discussion on the Aber Comp Sci Facebook group the day before this assignment's deadline, and did not want to re-do my SQL and potentially break anything but this is something I am now aware of and if I was to implement this for real I would implement an enumerated type for training_day so users could only enter valid days of the week.

```
drop table if exists members;
drop table if exists sports_participation;
drop table if exists team;
drop table if exists training;
drop table if exists team_players;

create table members (
  member_number text primary key,
  member_name text,
  member_phone text
);

insert into members(member_number, member_name, member_phone) values
  ('122', 'A Edwards', '07824079428'),
  ('107', 'J Stone', '07674873098'),
  ('193', 'E Simmonds', '07293709784'),
  ('752', 'TJ Jones', '07777123456'),
  ('773', 'TE Jones', '07398407198'),
  ('079', 'K Franks', '07937047823'),
  ('420', 'S Dogg', '07069126473'),
  ('230', 'K West', '07890234783'),
  ('123', 'S Carter', '0724808934'),
  ('359', 'K Lamar', '07234894012'),
  ('098', 'J Cole', '07823099021'),
  ('481', 'T Bangalter', '07001230941'),
  ('789', 'I Australia', '07666102938'),
  ('476', 'L Knope', '07423099758'),
  ('043', 'A Dwyer', '07883049821'),
  ('765', 'R Swanson', '07098123753'),
  ('254', 'J Winger', '07234784921');

create table sports_participation (
  member_number text references members(member_number),
  sport text,
  primary key (member_number, sport)
);

insert into sports_participation(member_number, sport) values
  ('122', 'Soccer'),
  ('122', 'Tiddlywinks'),
  ('122', 'Baseball'),
  ('107', 'Soccer'),
  ('193', 'Soccer'),
  ('752', 'Soccer'),
  ('773', 'Soccer'),
  ('079', 'Tiddlywinks'),
  ('420', 'Rugby'),
  ('420', 'Lacrosse'),
  ('420', 'Tiddlywinks'),
  ('420', 'Basketball'),
  ('230', 'Chess'),
  ('230', 'Tiddlywinks'),
  ('123', 'Hockey'),
  ('123', 'Tiddlywinks'),
  ('123', 'Table Tennis'),
  ('123', 'Basketball'),
  ('359', 'Rugby'),
  ('098', 'Rugby'),
  ('481', 'Baseball'),
  ('789', 'Basketball'),
  ('789', 'Baseball'),
  ('476', 'Basketball'),
  ('476', 'Cricket'),
  ('043', 'Basketball'),
  ('765', 'Basketball'),
  ('765', 'Water Polo'),
  ('254', 'Basketball');

create table team (
  team_name text primary key,
  sport text,
  captain_number text
);

insert into team(team_name, sport, captain_number) values
('Thunderbirds', 'Soccer', '752'),
('Aardvarks', 'Tiddlywinks', '079'),
('Warriors', 'Rugby', '359'),
('Red Sox', 'Baseball', '481'),
('Lakers', 'Basketball', '789'),
('Nets', 'Basketball', '476');

create table training (
  team_name text references team(team_name),
  training_day text,
  training_time time,
  primary key (team_name, training_day, training_time)
);

insert into training(team_name, training_day, training_time) values
  ('Thunderbirds', 'Tuesday', '7:30 pm'),
  ('Thunderbirds', 'Friday', '7:30 pm'),
  ('Aardvarks', 'Monday', '2:30 pm'),
  ('Warriors', 'Wednesday', '4:20 pm');

create table team_players (
  team_name text references team(team_name),
  member_number text references members(member_number),
  primary key (team_name, member_number)
);

insert into team_players(team_name, member_number) values
  ('Thunderbirds', '122'),
  ('Thunderbirds', '107'),
  ('Thunderbirds', '193'),
  ('Thunderbirds', '752'),
  ('Thunderbirds', '773'),
  ('Aardvarks', '122'),
  ('Aardvarks', '079'),
  ('Aardvarks', '420'),
  ('Aardvarks', '230'),
  ('Aardvarks', '123'),
  ('Warriors', '359'),
  ('Warriors', '420'),
  ('Warriors', '098'),
  ('Red Sox', '481'),
  ('Red Sox', '789'),
  ('Red Sox', '122'),
  ('Lakers', '789'),
  ('Lakers', '123'),
  ('Lakers', '420'),
  ('Nets', '476'),
  ('Nets', '043'),
  ('Nets', '765'),
  ('Nets', '254'),
  ('Nets', '098');
```