CS27020 Modeling Persistent Data

Llandwp Sports

Author	Max Atkins
Date Published	February 27, 2015
Department	Computer Science
Address	Aberystwyth University Penglais Campas Ceredigion SY23 3DB

Contents

1 Normalizing the Team Card **3**
1.1 Keys . . . 3
1.2 Un-Normalized . . . 3
1.3 Functional Dependencies . . . 4
1.4 1st Normal Form . . . 5
1.5 2nd Normal Form . . . 6
1.6 3rd Normal Form . . . 7

2 Normalizing the Member Card **8**
2.1 Un-Normalized . . . 8
2.2 Functional Dependencies . . . 8
2.3 1st Normal Form . . . 9
2.4 2nd Normal Form . . . 10
2.5 3rd Normal Form . . . 10

3 Optimising the Schema **11**
3.1 Creating the Optimised Normalisation . . . 11
3.2 Entity Relationship Diagram . . . 12

4 SQL Queries **13**
4.1 Table Creation Queries . . . 13
4.2 Insert Queries . . . 15
4.3 Select Queries . . . 20
4.3.1 List all players for a given team . . . 20
4.3.2 List all players for a given sport . . . 21
4.3.3 List all sports with no assigned teams . . . 21
4.3.4 List all sports with no assigned players . . . 21

5 Screenshots using PHPPGAdmin **21**

1 Normalizing the Team Card

1.1 Keys

Primary Key

Candidate Key

Foreign Key*

1.2 Un-Normalized

To create my un-normalized forms for the Team Card, I analyzed the it to identify all possible attributes, primary keys and repeating groups of attributes. I found that TeamName was an ideal Candidate key, but was not, in itself, enough to identify a team since two teams with the same name can play separate sports.

To solve this issue, combining TeamName and SportName into a composite primary key will ensure that TeamName and SportName are unique, but there can be multiple TeamNames.

I also identified candidate keys as potentials for unique identification for other tables: CaptainID, PlayerID and TrainingID. CaptainID and PlayerID appear to be very similar, however and will be dealt with later.

When looking for functional dependencies, I realized that TrainingDay and TrainingTime would require a unique ID to be assigned to a team.

I identified all attributes and found two repeating groups: Players and Trainings. This gave me this un-normalized form:

Teams(
    TeamName
    SportName
    CaptainID
    CaptainName
    CaptainPhone
    PlayerID
    PlayerName
    TrainingID
    TrainingDay
    TrainingTime
)
(PlayerID, PlayerName) Repeat for each team
(TrainingID, TrainingDay, TrainingTime) Repeat for each team

1.3 Functional Dependencies

To identify the functional dependencies, I analyzed the attributes to identify which unique values would depend on others. I identified the following functional dependencies:

TeamName and SportName, together, determine the Captain information, as these are the only attributes unique to a team. TeamName and SportName are grouped together, as specified above.
(TeamName, SportName) ->(CaptainID, CaptainName, CaptainPhone)

CaptainID determines Captain Information, but is also determined in the above functional dependency. This creates a transitive dependency and must be dealt with later. A Captain can not be uniquely identified by their name or phone number, but can be identified by a unique ID, hence the following dependency.
(CaptainID) ->(CaptainName, CaptainPhone)

A Player can not be uniquely determined by their name, but they can be determined by their ID number.
(PlayerID) ->(PlayerName)

TrainingDay and TrainingTime do not determine each other. However, I felt that a dependency was necessary to exist here to create an efficient table. To solve this, I created the TrainingID attribute, creating a TimeSlot concept to determine a combination of TrainingDay and TrainingTime.
(TrainingID) ->(TrainingDay, TrainingTime)

1.4 1st Normal Form

Proceeding to 1st normal form involves splitting off the repeating groups of attributes into their own tables. This involved creating foreign keys and identifying new primary keys for each new table. Since PlayerID determines PlayerName, it becomes the primary key for the Players table and since TrainingID determines TrainingDay and TrainingTime, that becomes the primary key for the TimeSlot table. I identified the following tables:

Teams(
    TeamName
    SportName
    CaptainID
    CaptainName
    CaptainPhone
)

Players(
    PlayerID
    PlayerName
    TeamName*
    SportName*
)

TimeSlot(
    TrainingID
    TrainingDay
    TrainingTime
    TeamName*
    SportName*
)

1.5 2nd Normal Form

Proceeding to 2nd normal form involves splitting off the partial dependencies. This involves creating a linking table between Teams and Players, since they are a many-to-many relationship (*..*). Since TimeSlot and Teams are a one-to-many relationship (a team has many timeslots, but a timeslot has one team), a direct relationship is enough and the table was unchanged from 1st normalisation.

At this stage, I also notice that, to ensure a player is unique with their team, all three attributes in the linking table must be a composite primary key.

I identified the following tables:

Teams(
    TeamName
    SportName
    CaptainID
    CaptainName
    CaptainPhone
)

HasPlayers(
    TeamName*
    SportName*
    PlayerID*
)

Players(
    PlayerID
    PlayerName
)

TimeSlot(
    TrainingID
    TrainingDay
    TrainingTime
    TeamName*
    SportName*
)

1.6 3rd Normal Form

Proceeding to 3rd normal form involves splitting off transitive dependencies. I only have one of these, and since a Captain is essentially a Player, the CaptainID can be linked directly to the Players table to retrieve a single player.

I identified the following tables:

Teams(
    TeamName
    SportName
    CaptainID*
)

HasPlayers(
    TeamName*
    SportName*
    PlayerID*
)

Players(
    PlayerID
    PlayerName
)

TimeSlot(
    TrainingID
    TrainingDay
    TrainingTime
    TeamName*
    SportName*
)

2 Normalizing the Member Card

2.1 Un-Normalized

To create my un-normalized forms for the Member Card, I followed the same steps as used for the previous card. I identified a more complicated set of repeating attributes, however. Each Member has a repeating SportName attribute, and for each repeating SportName, a repeating TeamName occurs. This nested repeating group needs to be split in turn from the inside out.

This gave me this un-normalized form:

Teams(
MemberNumber
MemberName
MemberPhone
SportName
TeamName
)
(SportName) Repeats for each member
(TeamName) Repeats for each sport a member plays

2.2 Functional Dependencies

To identify the functional dependencies, I, again, followed the same steps as above and found only one. I identified the following functional dependencies from this:

A member can not be uniquely determined by their name or phone number, but can be determined by their unique ID.
(MemberNumber) ->(MemberName, MemberPhone)

2.3 1st Normal Form

Proceeding to 1st normal form, I split off the nested repeating groups into their own tables, starting from the inner group. I also identified new primary and foreign keys.

I identified the following tables:

Members(
    MemberNumber
    MemberName
    MemberPhone
)

Sports(
    SportID
    SportName
    MemberNumber*
)

Teams(
    TeamName
    SportID*
)

2.4 2nd Normal Form

Proceeding to 2nd normal form, I separated Sports into a relation table for the many-to-many relationship.

Members(
    MemberNumber
    MemberName
    MemberPhone
)

PlayersSports(
    SportID*
    MemberNumber*
)

Sports(
    SportID
    SportName
    MemberNumber*
)

Teams(
    TeamName
    SportID*
)

2.5 3rd Normal Form

3rd normal form is the same as 2nd.

3 Optimising the Schema

3.1 Creating the Optimised Normalisation

To optimize the schema, I analysed both normalised cards to identify similar tables, and merged them, ensuring the relations were upheld and everything didn't get confusing. I identified that the Two Teams tables should be merged, and that the Player and Member tables should be merged. The relation for Sports and TimeSlots was kept for the Teams table. SportName become a foreign key to accommodate this change. I also created a relation between HasPlayers and PlayersSports to ensure that a player must play the sport BEFORE joining a team for that sport. This makes sense in a real life context.

I identified the following tables:

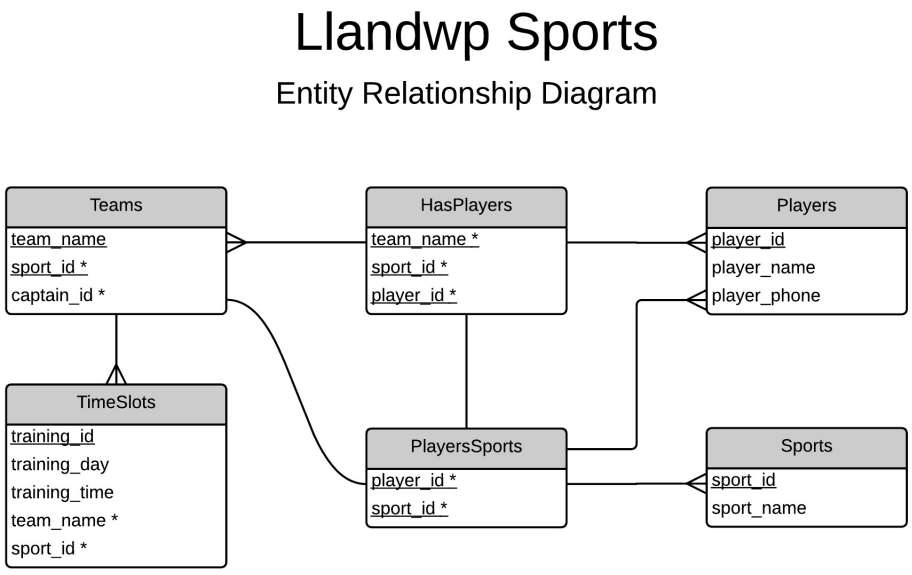
Teams(
    TeamName
    SportName*
    CaptainID*
)

HasPlayers(
    TeamName*
    SportName*
    PlayerID*
)

Players(
    PlayerID
    PlayerName
)

TimeSlot(
    TrainingID
    TrainingDay
    TrainingTime
    TeamName*
    SportName*
)
Sports(
    SportID
    SportName
)
PlayersSports(
    PlayerID*
    SportName*
)

3.2 Entity Relationship Diagram



4 SQL Queries

4.1 Table Creation Queries

Listed are my table creation queries that implement the schema detailed in the previous section. I used VARCHAR to hold text 20: characters for team_name, 40 characters for player_name and 15 characters for phone_number seem reasonable. I used the SERIAL type to create auto incrementing primary keys. The INT type is used for non-primary key IDs (Eg. foreign keys). I also create my own enumerated (ENUM) type to hold a listing of the days, to ensure that only the actual days can be inputted to the database.

```
/*This is my sports table, primary key sport_id,
  to hold a list of all sports at the club
*/
create table IF NOT EXISTS sports (
        sport_id SERIAL,
        sport_name VARCHAR(20),
        PRIMARY KEY(sport_id)
);

/*This is my players table, primary key player_id,
  to hold a list of all players at the club
*/
create table IF NOT EXISTS players (
        player_id SERIAL,
        player_name VARCHAR(40),
        player_phone VARCHAR(15),
        PRIMARY KEY(player_id)
);

/*This is my players_sports table, primary composite key player_id,
  sport_id; foreign keys player_id, sport_id from the players and
  sports tables. This is my linking table to establish the many-to-many
  relationship between players and sports. It assigns a sport to a player,
  without a team. This is also where the teams table takes it's sport
  and captain.
*/
create table IF NOT EXISTS players_sports (
        player_id INT REFERENCES players(player_id),
        sport_id INT REFERENCES sports(sport_id),
        PRIMARY KEY (player_id, sport_id)
);
```

```
/*This is my teams table, primary composite key team_name, sport_id;
  foreign key sport_id, captain_id from the players_sports table.
  This table holds all the teams at the club,  the sport it plays,
  and it's captain
*/
create table IF NOT EXISTS teams (
        team_name VARCHAR(20),
        sport_id INT,
        captain_id INT,
        FOREIGN KEY (sport_id, captain_id)
                REFERENCES players_sports(sport_id, player_id),
        PRIMARY KEY (team_name, sport_id)

);
/*This is my has_players table, primary composite key team_name, sport_id,
  player_id; foreign keys team_name, sport_Id from the teams table. This is
  my linking table to establish the many-to-many relationship between teams
  and players.
*/
create table IF NOT EXISTS has_players (
        team_name VARCHAR(20),
        sport_id int,
        player_id INT REFERENCES players(player_id),
        FOREIGN KEY (team_name, sport_id)
                REFERENCES teams(team_name, sport_id),
        FOREIGN KEY (player_id, sport_id)
                REFERENCES players_sports(player_id, sport_id),
        PRIMARY KEY(team_name, sport_id, player_id)
);

/*This is the enumerated DAY type I created which holds all the days
  a training_day can hold.
*/
CREATE TYPE DAY AS ENUM (
        'Monday',
        'Tuesday',
        'Wednesday',
        'Thursday',
        'Friday',
        'Saturday',
        'Sunday'
);
```

```
/*This is my time_slot table, primary key training_ID; foreign keys
  team_name, sport_id from the teams table to establish the
  one-to-many relationship. This table holds all the training day/time
  data for each team.
*/
create table IF NOT EXISTS time_slot (
        training_id SERIAL,
        training_day DAY,
        training_time TIME,
        team_name VARCHAR(20),
        sport_id int,
        FOREIGN KEY (team_name, sport_id)
                REFERENCES teams (team_name, sport_id),
        PRIMARY KEY (training_id)
);
```

4.2 Insert Queries

```
/*Insert 7 players into the players table.*/

INSERT INTO players (player_name, player_phone)
        VALUES ('Max', '07949028660');
INSERT INTO players (player_name, player_phone)
        VALUES ('Hannah', '07945624312');
INSERT INTO players (player_name, player_phone)
        VALUES ('Chandlar', '07947687776');
INSERT INTO players (player_name, player_phone)
        VALUES ('Edward', '07945432123');
INSERT INTO players (player_name, player_phone)
        VALUES ('Liam', '07945678765');
INSERT INTO players (player_name, player_phone)
        VALUES ('Erin', '07949996758');
INSERT INTO players (player_name, player_phone)
        VALUES ('Ffion', '07946576566');

/*Insert 4 sports into the sports table.*/
INSERT INTO sports (sport_name)
        VALUES ('Soccer');
INSERT INTO sports (sport_name)
        VALUES ('Table_Tennis');
INSERT INTO sports (sport_name)
        VALUES ('Chess');
INSERT INTO sports (sport_name)
        VALUES ('Squash');
```

```
INSERT INTO sports (sport_name)
        VALUES ('Rugby');


/*Assign sports to players with no teams.*/
/*Max plays Soccer*/
INSERT INTO players_sports(player_id, sport_id)
        VALUES (1, 1);
/*Max plays Table Tennis.*/
INSERT INTO players_sports(player_id, sport_id)
        VALUES (1, 2);
/*Hannah plays Soccer.*/
INSERT INTO players_sports(player_id, sport_id)
        VALUES (2, 1);
/*Hannah plays Squash.*/
INSERT INTO players_sports(player_id, sport_id)
        VALUES (2, 4);
/*Chandlar plays Soccer.*/
INSERT INTO players_sports(player_id, sport_id)
        VALUES (3, 1);
/*Chandlar plays Chess*/
INSERT INTO players_sports(player_id, sport_id)
        VALUES (3, 3);
/*Chandlar plays Squash.*/
INSERT INTO players_sports(player_id, sport_id)
        VALUES(3, 4);
/*Edward plays Soccer.*/
INSERT INTO players_sports(player_id, sport_id)
        VALUES (4, 1);
/*Edward plays Table Tennis.*/
INSERT INTO players_sports(player_id, sport_id)
        VALUES (4, 2);
/*Liam plays Soccer.*/
INSERT INTO players_sports(player_id, sport_id)
        VALUES (5, 1);
/*Liam plays Chess.*/
INSERT INTO players_sports(player_id, sport_id)
        VALUES (5, 3);
/*Erin plays Soccer.*/
INSERT INTO players_sports(player_id, sport_id)
        VALUES (6, 1);
/*Ffion plays Table Tennis.*/
INSERT INTO players_sports(player_id, sport_id)
        VALUES (7, 2);
/*Ffion plays Chess.*/
INSERT INTO players_sports(player_id, sport_id) VALUES (7, 3);
```

```
/*Insert 4 teams into the teams table, assigning captains and
  sports to each.
*/
/*Hannah is the captain of Thunderbirds; they play Soccer.*/
INSERT INTO teams (team_name, sport_id, captain_id)
        VALUES ('Thunderbirds', 1, 2);
/*Chandlar is the captain of Flying Machine; they play Soccer.*/
INSERT INTO teams (team_name, sport_id, captain_id)
        VALUES ('Flying Machine', 1, 3);
/*Ffion is the captain of The Tables; they play Table Tennis.*/
INSERT INTO teams (team_name, sport_id, captain_id)
        VALUES ('The Tables', 2, 7);
/*Liam is the captain of The Chessers; they play Chess.*/
INSERT INTO teams (team_name, sport_id, captain_id)
        VALUES ('The Chessers', 3, 5);

/*Broken teams query to show a captain must play the sport first*/
INSERT INTO teams (team_name, sport_id, captain_id)
        VALUES ('The Chessers', 3, 5);
```

```
/*Assign players to their teams*/
/*Max plays with Thunderbirds.*/
INSERT INTO has_players (team_name, sport_id, player_id)
        VALUES ('Thunderbirds', 1, 1);
/*Liam plays with Thunderbirds.*/
INSERT INTO has_players (team_name, sport_id, player_id)
        VALUES ('Thunderbirds', 1, 5);
/*Erin plays with Thunderbirds.*/
INSERT INTO has_players (team_name, sport_id, player_id)
        VALUES ('Thunderbirds', 1, 6);
/*Edward plays with Flying Machine.*/
INSERT INTO has_players (team_name, sport_id, player_id)
        VALUES ('Flying Machine', 1, 4);
/*Edward plays with The Tables.*/
INSERT INTO has_players (team_name, sport_id, player_id)
        VALUES ('The Tables', 2, 4);
/*Max plays with The Tables.*/
INSERT INTO has_players (team_name, sport_id, player_id)
        VALUES ('The Tables', 2, 1);
/*Chandlar plays with The Chessers.*/
INSERT INTO has_players (team_name, sport_id, player_id)
        VALUES ('The Chessers', 3, 3);
/*Ffion plays with The Chessers.*/
INSERT INTO has_players (team_name, sport_id, player_id)
        VALUES ('The Chessers', 3, 7);

/*Broken has_players query to prove a player must play a
  sport before being a part of the team
  */
INSERT INTO has_players (team_name, sport_id, player_id)
        VALUES ('The Chessers', 4, 1);

/*Assign training days and times to each team.*/
INSERT INTO time_slot (training_day, training_time, team_name, sport_id)
        VALUES ('Tuesday', '09:30', 'Thunderbirds', 1);
INSERT INTO time_slot (training_day, training_time, team_name, sport_id)
        VALUES ('Thursday', '14:30', 'Thunderbirds', 1);
INSERT INTO time_slot (training_day, training_time, team_name, sport_id)
        VALUES ('Monday', '10:00', 'Flying Machine', 1);
INSERT INTO time_slot (training_day, training_time, team_name, sport_id)
        VALUES ('Sunday', '04:00', 'The Tables', 2);
INSERT INTO time_slot (training_day, training_time, team_name, sport_id)
        VALUES ('Monday', '12:00', 'The Tables', 2);
```

```
INSERT INTO time_slot (training_day, training_time, team_name, sport_id)
        VALUES ('Monday', '14:00', 'The Tables', 2);
INSERT INTO time_slot (training_day, training_time, team_name, sport_id)
        VALUES ('Wednesday', '16:00', 'The Chessers', 3);

/*Broken time_slot query to prove ENUM works*/
INSERT INTO time_slot (training_day, training_time, team_name, sport_id)
        VALUES ('Fednesdays', '16:00', 'The Chessers', 3);
```

4.3 Select Queries

4.3.1 List all players for a given team

```
/*List all players on the Thunderbirds team.
  Should be Max, Liam, Erin.
*/
SELECT players.player_name, players.player_phone FROM has_players
        JOIN players ON has_players.player_id = players.player_id
        WHERE has_players.team_name = 'Thunderbirds'
        AND has_players.sport_id = 1;

/*List all players on the Flying Machine team.
  Should be Edward.
*/
SELECT players.player_name, players.player_phone FROM has_players
        JOIN players on has_players.player_id = players.player_id
        WHERE has_players.team_name = 'Flying Machine'
        AND has_players.sport_id = 1;

/*List all players on The Tables team.
  Should be Max, Edward.
*/
SELECT players.player_name, players.player_phone FROM has_players
        JOIN players ON has_players.player_id = players.player_id
        WHERE has_players.team_name = 'The Tables'
        AND has_players.sport_id = 2;

/*List all players on The Chessers team.
  Should be Chandlar, Ffion.
*/
SELECT players.player_name, players.player_phone FROM has_players
        JOIN players ON has_players.player_id = players.player_id
        WHERE has_players.team_name = 'The Chessers'
        AND has_players.sport_id = 3;
```

4.3.2 List all players for a given sport

```
/*List all players who play Soccer.
  Should be Max, Hannah, Chandlar, Edward, Liam, Erin
*/
SELECT sports.sport_name, players.player_name FROM players_sports
        JOIN players ON players_sports.player_id = players.player_id
        JOIN sports ON  sports.sport_id = players_sports.sport_id
        WHERE sports.sport_name  = 'Soccer';

/*List all players who play Table Tennis.
  Should be Max, Edward, Ffion.
*/
SELECT sports.sport_name, players.player_name FROM players_sports
        JOIN players ON players_sports.player_id = players.player_id
        JOIN sports ON  sports.sport_id = players_sports.sport_id
        WHERE sports.sport_name  = 'Table Tennis';

/*List all players who play Chess.
  Should be Chandlar, Liam, Ffion.
*/
SELECT sports.sport_name, players.player_name FROM players_sports
        JOIN players ON players_sports.player_id = players.player_id
        JOIN sports ON  sports.sport_id = players_sports.sport_id
        WHERE sports.sport_name  = 'Chess';

/*List all players who play Squash.
  Should be Hannah.
*/
SELECT sports.sport_name, players.player_name FROM players_sports
        JOIN players ON players_sports.player_id = players.player_id
        JOIN sports ON  sports.sport_id = players_sports.sport_id
        WHERE sports.sport_name  = 'Squash';
```

4.3.3 List all sports with no assigned teams

```
SELECT sports.sport_name FROM sports
        LEFT JOIN teams ON teams.sport_id = sports.sport_id
        WHERE teams.sport_id IS NULL;
```

4.3.4 List all sports with no assigned players

```
SELECT sports.sport_name FROM sports
        LEFT JOIN players_sports ON players_sports.sport_id = sports.sport_id
        WHERE players_sports.sport_id IS NULL;
```

5 Screenshots using PHPPGAdmin

No tables found.

Create table

Figure 1: Empty database.

	Table	Owner	Tablespace	Estimated row count	Actions									Comment
☐	has_players	mta2		0	Browse	Select	Insert	Empty	Alter	Drop	Vacuum	Analyze	Reindex	
☐	players	mta2		0	Browse	Select	Insert	Empty	Alter	Drop	Vacuum	Analyze	Reindex	
☐	players_sports	mta2		0	Browse	Select	Insert	Empty	Alter	Drop	Vacuum	Analyze	Reindex	
☐	sports	mta2		0	Browse	Select	Insert	Empty	Alter	Drop	Vacuum	Analyze	Reindex	
☐	teams	mta2		0	Browse	Select	Insert	Empty	Alter	Drop	Vacuum	Analyze	Reindex	
☐	time_slot	mta2		0	Browse	Select	Insert	Empty	Alter	Drop	Vacuum	Analyze	Reindex	

Actions on multiple lines

Select all / Unselect all ---> Analyze ▾ Execute

Figure 2: Used creation queries listed above.

Actions		player_id	player_name	player_phone
Edit	Delete	1	Max	07949028660
Edit	Delete	2	Hannah	07945624312
Edit	Delete	3	Chandlar	07947687776
Edit	Delete	4	Edward	07945432123
Edit	Delete	5	Liam	07945678765
Edit	Delete	6	Erin	07949996758
Edit	Delete	7	Ffion	07946576566

Figure 3: Inserted players.

Actions		sport_id	sport_name
Edit	Delete	1	Soccer
Edit	Delete	2	Table Tennis
Edit	Delete	3	Chess
Edit	Delete	4	Squash
Edit	Delete	5	Rugby

Figure 4: Inserted sports.

Actions		player_id	sport_id
Edit	Delete	1	1
Edit	Delete	1	2
Edit	Delete	2	1
Edit	Delete	2	4
Edit	Delete	3	1
Edit	Delete	3	3
Edit	Delete	4	1
Edit	Delete	4	2
Edit	Delete	5	1
Edit	Delete	5	3
Edit	Delete	6	1
Edit	Delete	7	2
Edit	Delete	7	3

Figure 5: Inserted player-sports relations.

Actions		team_name	sport_id	captain_id
Edit	Delete	Thunderbirds	1	2
Edit	Delete	Flying Machine	1	3
Edit	Delete	The Tables	2	7
Edit	Delete	The Chessers	3	5

Figure 6: Inserted teams with captains.

Actions		team_name	sport_id	player_id
Edit	Delete	Thunderbirds	1	1
Edit	Delete	Thunderbirds	1	5
Edit	Delete	Thunderbirds	1	6
Edit	Delete	Flying Machine	1	4
Edit	Delete	The Tables	2	4
Edit	Delete	The Tables	2	1

Figure 7: Inserted teams-players relations.

Actions		training_id	training_day	training_time	team_name	sport_id
Edit	Delete	1	Tuesday	09:30:00	Thunderbirds	1
Edit	Delete	2	Thursday	14:30:00	Thunderbirds	1
Edit	Delete	3	Monday	10:00:00	Flying Machine	1
Edit	Delete	4	Sunday	04:00:00	The Tables	2
Edit	Delete	5	Monday	12:00:00	The Tables	2
Edit	Delete	6	Monday	14:00:00	The Tables	2

Figure 8: Inserted timeslots for teams.

player_name	player_phone
Max	07949028660
Liam	07945678765
Erin	07949996758

Figure 9: Players who play for Thunderbirds.

sport_name	player_name
Soccer	Max
Soccer	Hannah
Soccer	Chandlar
Soccer	Edward
Soccer	Liam
Soccer	Erin

Figure 10: Players who play Soccer.

sport_name
Squash
Rugby

Figure 11: Sports with no teams

sport_name
Rugby

Figure 12: Sports with no players.

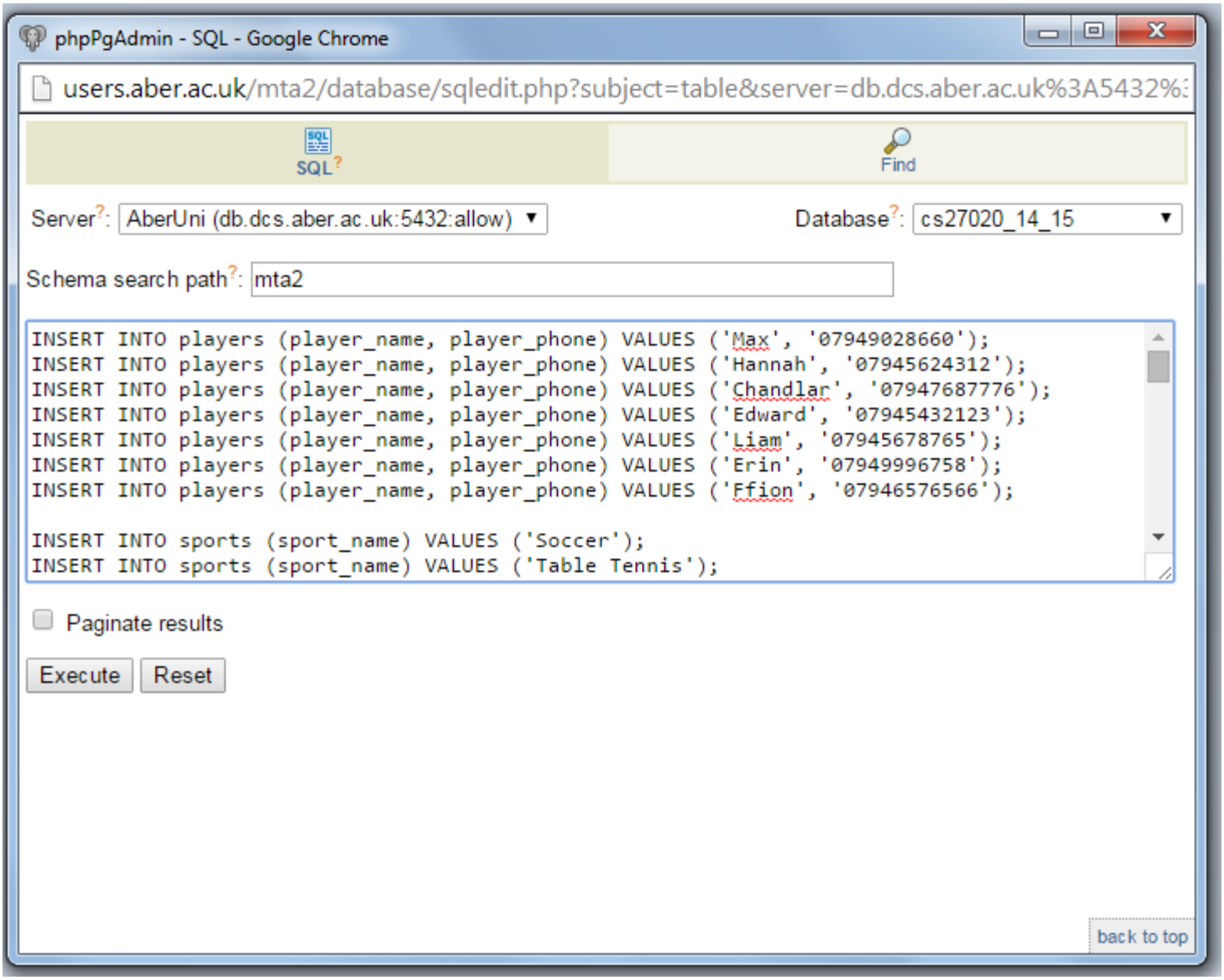


Figure 13: How I executed my queries, not using provided tools.