Killian Rutherford (KRR2125)
Richard Godden (RG3047) COMS 4111 Project 1.1

# Tennis Singles Tournament Database

**Approved By Aayush Mudgal**

## Paragraph

1. The database will be storing information for singles tennis tournaments, in the point of view of a tournament organiser/data provider. There will be seven entities (revised from the original 6 we brought to the meeting): Tournaments, Players, Matches, Courts, Spectators, Tickets and the newly added Complex. The organisation aspect will involve court numbers/tickets and spectators, whilst data provided will be relevant to player and match stats. Users will be able to query this database to discover player stats and match outcomes (E.g. who won which match, what those match statistics were, what the player statistics in the tournament were, what their overall statistics are, etc..). As well as looking at logistics, enquiries about where the matches are taking place, ticket prices, and spectator statistics. Challenges will involve the large number of features we will have to include in this database, as well as looking at the relationships between the entity sets. Some are not so obvious, particularly with the tickets (can be different types), not necessarily one to one relationships (1 ticket may not be to 1 match) and name changes etc. For now we have restricted this condition to one match per ticket which, we may change in the future.

   After meeting with the TA, we made the following revisions:

   (a) We incorporated a few more real-world constraints (see below)

   (b) We modified our E-R diagram as we initially had too many complex attributes for each set

   (c) We changed relationships between sets (in matches v. players case)

   (d) We modified our SQL schema so that we did not necessarily have one table for each relational set. In the 1-1 and 1-many cases we just included a foreign key within the respective entity table

We have factored in real-world constraints throughout. The most predominant examples are our SQL constraints, such as ensuring there is a unique winner for each match. Further, a match can have multiple start times, which would indicate a suspension (and may end up being played on different courts). In the examples of tickets, we are also distinguishing buyers from reserved for, as 1 person can buy multiple tickets for others. Many more of real world relationships can be observed on our E-R diagram.

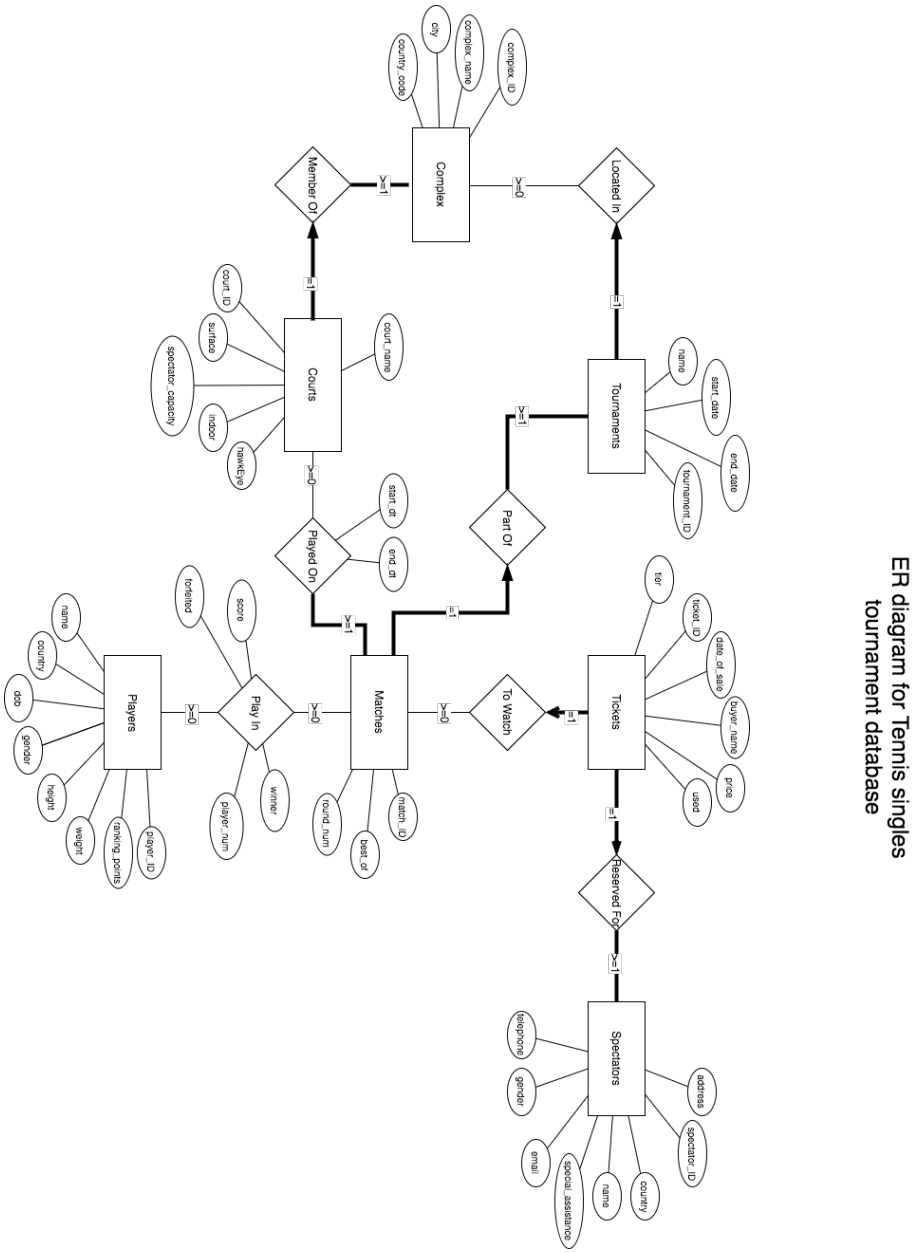
We will be choosing the Web Front-End Option for Part 3 (Option 3.a).

We will fill all the player and match data with real data from Jeff Sackmann tennis_atp github ($https://github.com/JeffSackmann/tennis\_atp$). The spectator and ticket details may require further research /fill in fake data.

### Contingency Plan

We do not anticipate needing this, but our contingency plan would simply be to take out the Tickets and Spectators entity sets, and simply deal with Players, Matches, Complex, Courts and Tournaments. After our meeting, the TA told us we could even further eliminate one of the sets Courts or Complex if need be.

## E-R Diagram


ER diagram for Tennis singles tournament database
Complex
city
complex_name
complex_ID
country_code
Member Of
Located In
Courts
court_ID
surface
spectator_capacity
indoor
hawkEye
court_name
Tournaments
name
start_date
end_date
tournament_ID
Played On
start_dt
end_dt
Part Of
Players
name
country
dob
gender
height
weight
ranking_points
player_ID
Play In
totalled
score
player_num
winner
Matches
round_num
best_of
match_ID
To Watch
Tickets
tier
ticket_ID
date_of_sale
buyer_name
price
used
Reserved For
Spectators
telephone
gender
email
special_assistance
name
country
address
spectator_ID

### SQL Schema

```
CREATE TABLE Tournaments
       (tournament_ID INTEGER,
        name VARCHAR NOT NULL,
        start_date DATE NOT NULL,
        end_date DATE,
        complex_ID INTEGER NOT NULL,
        FOREIGN KEY (complex_ID) REFERENCES Complex,
                ON DELETE NO ACTION
                ON UPDATE CASCADE
        PRIMARY KEY (tournament_ID))

CREATE TABLE Matches
       (match_ID INTEGER,
        best_of INTEGER NOT NULL,
        round_num INTEGER NOT NULL,
        tournament_ID INTEGER NOT NULL,
        FOREIGN KEY (tournament_ID) REFERENCES Tournaments,
                ON DELETE CASCADE
                ON UPDATE CASCADE
        PRIMARY KEY (match_ID))

CREATE TABLE Played_On
       (court_ID INTEGER,
        match_ID INTEGER,
        start_dt DATE-TIME NOT NULL,
        end_dt DATE-TIME,
        PRIMARY KEY (match_ID,court_ID,start_dt),
        FOREIGN KEY (court_ID) REFERENCES Courts,
                ON DELETE DO NOTHING
                ON UPDATE CASCADE
        FOREIGN KEY (match_ID) REFERENCES Matches,
                ON DELETE CASCADE
                ON UPDATE CASCADE)

CREATE TABLE Courts
       (court_ID INTEGER,
        court_name VARCHAR NOT NULL,
        surface VARCHAR,
```

```
        spectator_capacity INTEGER,
        indoor BOOLEAN NOT NULL,
        hawkEye BOOLEAN NOT NULL,
        complex_ID INTEGER NOT NULL,
        FOREIGN KEY (complex_ID) REFERENCES Complex,
                ON DELETE NO ACTION
                ON UPDATE CASCADE
        PRIMARY KEY (court_ID),
        UNIQUE (complex_ID, court_name))

CREATE TABLE Complex
       (complex_ID INTEGER,
        complex_name VARCHAR NOT NULL,
        city VARCHAR,
        country CHAR(3),
        PRIMARY KEY (complex_ID))

CREATE TABLE Players
       (player_ID INTEGER,
name VARCHAR NOT NULL,
country CHAR(3),
dob DATE,
gender CHAR(1) NOT NULL,
height REAL,
weight REAL,
ranking_points INTEGER NOT NULL,
PRIMARY KEY (player_ID))

CREATE TABLE Play_In
       (player_ID INTEGER NOT NULL,
match_ID INTEGER NOT NULL,
        player_number INTEGER NOT NULL,
        winner BOOLEAN,
        forfeited BOOLEAN,
        score VARCHAR,
PRIMARY KEY (player_ID, match_ID),
FOREIGN KEY (player_ID) REFERENCES Players,
                ON DELETE NO ACTION
                ON UPDATE CASCADE
        FOREIGN KEY (match_ID) REFERENCES Matches,
```

```
                ON DELETE NO ACTION
                ON UPDATE CASCADE
        UNIQUE (match_ID, winner))

CREATE TABLE Tickets
       (ticket_ID INTEGER NOT NULL,
buyer_name VARCHAR NOT NULL,
price REAL NOT NULL,
date_of_sale DATE,
used BOOLEAN NOT NULL,
        tier VARCHAR,
        match_ID INTEGER NOT NULL,
        spectator_ID INTEGER NOT NULL,
        FOREIGN KEY (spectator_ID) REFERENCES Spectators,
                ON DELETE NO ACTION
                ON UPDATE CASCADE
        FOREIGN KEY (match_ID) REFERENCES Matches,
                ON DELETE NO ACTION
                ON UPDATE CASCADE
PRIMARY KEY (ticket_ID)
        UNIQUE (spectator_ID,match_ID))

CREATE TABLE Spectators
       (spectator_ID INTEGER NOT NULL,
name VARCHAR NOT NULL,
address VARCHAR,
country CHAR(3),
email VARCHAR,
telephone INTEGER,
gender CHAR(1),
special_assistance BOOLEAN,
PRIMARY KEY(spectator_ID))
```

## Copy of Proposal approved by TA

**Approved By Aayush Mudgal**

Paragraph:
The database will be storing information for singles tennis tournaments, in the point of view of a tournament organiser/data provider. There will be six entities: Tournament, Players, Matches, Stadiums/Courts, Spectators and Tickets.
The organisation aspect will involve court numbers/tickets and spectators, whilst data provided will be relevant to player and match stats.
We will be chosing the Web Front-End Option for Part 3. Users will be able to query this database to discover player stats and match outcomes (E.g. who won which match, what those match statistics were, what the player statistics in the tournament were, what their overall statistics are, etc..). As well as looking at logistics, enquiries about where the matches are taking place, ticket prices, and spectator statistics.
Challenges will involve the large number of features we will have to include in this database, as well as looking at the relationships between the entity sets. Some are not so obvious, particularly with the tickets (can be different types), not necessarily one to one relationships (1 ticket may not be to 1 match) and name changes etc. Additionally how to deal with suspended matches that possibly move courts/days.

We will fill all the player and match data with real data from Jeff Sackmann tennis_atp github. The spectator and ticket details may require further research /fill in fake data.

Contingency plan: Deal only with the players/matches/tournaments and maybe courts, as the data is already there in csv's.