# Assignment 1

Ahmad Baig

October 2019

**EECS: ahmad85**

**Student Number: 215180870**

**Number of Grace days Used 2 and a half**

# 1 Introduction

This assignment we will be writing queries in relational algebra on a data base using a schema which is already given to us. The schema that was given to us is concerning tennis and various relations regarding professional tennis played on an international scale.

# 2 Additional Integrity Constraints

Five integrity constraints have been given and I will define them using the notation from section 2.5.1 of the textbook.

## 2.1 Constraint 1: No player can play against themselves

This constraint will be shown by selecting Events where the playerA column and the playerB column are equivalent to one another and then showing that this relation is part of the empty set as shown below.

$$\sigma_{playerA=playerB}(Event) \subseteq \phi$$

## 2.2 Constraint 2: All vouchers for an event have to be purchased before the time of the event

For this constraint we will equijoin the Event and Voucher tables on EID and we will select only the tuples where the dateIssued is greater than the date of the event and we will show that the result of this relation is a subset of the empty set as can be seen below.

$$\sigma_{dateIssued>date}(Event \bowtie Voucher) \subseteq \phi$$

## 2.3 Constraint Three: the total number of vouchers purchased should not exceed the court capacity

It appears that this constraint cant be show in relational algebra due to the fact that in order to prevent more vouchers to be bought than the capacity of the event we would have to count all of the vouchers that have been bought and that is not possible in Relational algebra due to the lack of aggregate functions.

## 2.4 Constraint 4: There Exist players who have never played in any event

In this constraint we will equijoin players an event on pid=playerA and then we will union it with another equijoin betweeen the Player and Event tables so we get all of the player possible from that we will select only the columns corresponding to the players table and subtract it from every column in players and show that this is still a subset of players.

$(Player) - \Pi_{PID,fname,lname,gender,activeSince,wins,losses,totalPoints,globalRank,ctryID}$
$((Player \bowtie_{PID=playerA} Event) \cup (Player \bowtie_{PID=playerB} Event)) \subseteq Player$

### 2.5 Constraint 5: All tournament slams must be 'AO', 'FO', 'UO' or 'W'

In this constraint we will select all of the rows from the Tournament table that has the respective acronyms mentioned above an then we will subtract this relation from all possible Tournament values and showing that this should be apart of the empty set.

$$(Tournament) - \sigma_{slam='AO'='FO'='UO'='W'}(Tournament) \subseteq \phi$$

## 3 Relational Algebra Queries

In this section we will design queries using relational algebra from the given description of what the query should be. Explanations will also be provided.

### 3.1 Query 1: Report the first and last name of players the competed in every court if there are ties report them

In this query we find all of the players that played in ever court. First we find all of them for playerA and them we do the same for PlayerB and at the very end we join the together

**finding all for PlayerA**

$PE := \Pi_{PID,fname,lname} Player \bowtie_{PID=playerA} \Pi_{EID,playerA,courtID} Event$

$P1 := \Pi_{PID,fname,lname,EID,playerA} PE -$
$\Pi_{PID,fname,lname,EID,playerA}((\Pi_{PID,fname,lname,EID,playerA} PE \times \Pi_{CID} Court) -$
$PE)$

**finding the all for playerB**

$PL := \Pi_{PID,fname,lname} Player \bowtie_{PID=playerB} \Pi_{EID,playerB,courtID} Event$

$P2 := \Pi_{PID,fname,lname,EID,playerB} PL -$
$\Pi_{PID,fname,lname,EID,playerB}((\Pi_{PID,fname,lname,EID,playerB} PL \times \Pi_{CID} Court) -$
$PL)$

**Joing them together**

$\Pi_{fname,lname}(P1 \cup P2)$

## 3.2 Query 2: Report the EID of the event with the highest number of vouchers that were purchased

Unfortunately this query cannot be accomplished due to the fact that there is a lack of aggregation functions in relational algebra. In order to accomplish this we would need to add up all of the vouchers of each and every event and then we would have to compare them with one another but since we cannot add them all up this renders us incapable of accomplishing this query.

## 3.3 Query 3: Report names of various countries that have had players who didnt play in any event

In this query we will equijoin the Player and events table on PID is equivalent to playerA and then we will union that with another equijoin between Player and Event where this time PID is equivalent to playerB. We do this to get all the players that have played in one or more events then we project the columns from the player table that we need and subtract from all of the values in the Player table which allows us to have access to all of the player that haven't played in any event. Then we equijoin with countries and project the country name only.

$\Pi_{name}((Country) \bowtie_{CTRYID=ctryID} ((Player) -$
$\Pi_{PID,fname,lname,gender,activeSince,wins,losses,totalPoints,globalRank,ctryID}$
$((Player \bowtie_{PID=playerA} Event) \cup (Player \bowtie_{PID=playerB} Event))))$

## 3.4 Query 4: Report the CID(s) and the name of the court where exactly one event took place

In order to accomplish this query we will obtain the cases where their are more than 2 events at a cour and 0 events at a court and subtract them from the set of all courts to obtain the Court(s) that have only one event in them.

**R5 is the case where zero events took place in the Court:**
$R5 := \Pi_{CID,name}(Court) - Pi_{CID,name}((Court) \bowtie_{CID=courtID} (Event))$

**R1 and R2 are two cases where the Court table is joined with the Event table:**
$R1 := \Pi_{EID,CID,name}(Court) \bowtie_{CID=countryID} (Event)$
$R2 := \Pi_{EID,CID,name}(\rho_{EID,CID,name \rightarrow EID1,CID1,name1}((Court) \bowtie CID=countryID(Event)))$

**R1 and R2 are joined together to find the courts where 2 or more events took place by making sure their EID's are not equal but their CID's are:**
$R3 := \Pi_{CID,name}((R1) \bowtie_{EID \neq EID1 \, and \, CID=CID1} (R2))$

**We subtract the 2 or more case from the set of all possible Courts:**
$R4 := \Pi_{CID,name}(Court) - R3$

**We subtract the 0 case from the set of all possible courts:**
$R4 - R5$

## 3.5 Query 5: Report the countries of players with the highest difference in the number of sets won when competing with each other

We cannot report on this query because it would require for us to subtract values and store them in a new column and then compare them. Unfortunately, we can't subtract values from each other in Relational Algebra.

## 3.6 Query 6: Report the PID of the Players who have played in the courts with the largest capacity

In this query we will first find the court with the highest capacity and then join that with the Events table and the player table and extract the PID from that.

**first we join court with itself and get all the courts with some capcity less than the other courts:**
$C1 := Court$
$C2 := \Pi_{C1.capacity,C1.name,C1.CID}(Court \bowtie_{Court.capacity>C1.capacity} C1)$

**Then we subtract C2 from the set of all possible courts in order to get the on with the maximum capacity:**
$C3 := Court - C2$

**Then we join the court with highest capacity with the Events table and get the get the union of playerA and playerB and also join it with the player table and from that select the PID's:**
$\Pi_{PID}((C3 \bowtie_{courtID=CID} Event) \bowtie_{PID=playerA} Player)$
$\cup$
$\Pi_{PID}((C3 \bowtie_{courtID=CID} Event) \bowtie_{PID=playerB} Player)$

## 3.7 Query 7: Find the country that won the Event for which the very first voucher was bought

In this query we will first fin the oldest Voucher then we will find the event for that Voucher and after which we find the which Player won the game and join that that with both the Player table and then the Country table in order to project the winning countries name.

**First we theta join Voucher with itself where dateIssued is greater then the other Vouchers date Issued and then we project the values where the date Issued is greater than the other date Issued:**
$V1 := Voucher$
$V2 := \Pi_{V1.VID,V1.dateIssued,V1.EID}((V1) \bowtie_{V1.dateIssued>Voucher.dateIssued} (Voucher))$

**then we subtract the vouchers that have some date greater than another date in the set from the set of all Vouchers giving us the oldest voucher:**
$V3 := Voucher - V2$

**we join Voucher with Event:**
$VE := (V2) \bowtie (Event)$

**From this we obtain the both scenario where playerA won or PlayerB won and Union them:**
$PA := (\sigma_{setswonA>setwonB}(VE)) \bowtie_{playerA=PID} (Player)$
$PB := (\sigma_{setswonB>setwonA}(VE)) \bowtie_{playerB=PID} (Player)$
$PA1 := PA \cup PB$

**Then we join the previous relation with Country and project the country name:**
$\Pi_{name}(PA1 \bowtie_{ctryID=CTRYID} (Country))$

## 3.8 Query 8: Find the first and last name of the second highest total points from players of the same country

For this query We first find the Highest player then we subtract that from all the possible players to get all possible players without the highest and then we find the highest player in that set using the same method as previously used.
**We first find the highest player and Subtract him from the set of all possible Values:**
$P := (\sigma_{name='Canada'}(Country)) \bowtie (Player)$
$P1 := (\sigma_{name='Canada'}(Country)) \bowtie (Player)$

$P2 := \Pi_{P1.PID,P1.fname,P1.lname.P1.gender,P1.activeSince,P1.wins,P1.losses,P1.totalPoints,P1.globalRank,P1.ctryID}$
$((P) \bowtie_{P.totalPoints>P1.totalPoints} (P1))$

$P3 := Player - P2$

**Then we subtract that from the set of all players:**
$P4 := Player - P3$

**Then We repeat on the set of all players without the highest player**

$P5 := (\sigma_{name='Canada'}(Country)) \bowtie (P4)$
$P6 := (\sigma_{name='Canada'}(Country)) \bowtie (P4)$

$P7 := \Pi_{P6.PID,P6.fname,P6.lname,P6.gender,P6.activeSince,P6.wins,P6.losses,P6.totalPoints,P6.globalRank,P6.ctryID}$
$(((P5)) \bowtie_{P5.totalPoints>P6.totalPoints} (P6))$
$P4 - P7$

### 3.9 Query 9: Report the EID(s) of Event where at least 2 Vouchers were bought on the day of the Event

For this Query we cross join Voucher with Event on the dateIssued being the same as the date of the Event and then we join the previous relation With a similar one and and check that the event is the same but the VID's are different and from that we project the EID's
$V1 := (Voucher) \bowtie_{dateIssued=dateandVoucher.EID=Event.EID} (Event)$
$V2 := (Voucher) \bowtie_{dateIssued=dateandVoucher.EID=Event.EID} (Event)$
$\Pi_{V1.EID}((V1) \bowtie_{V1.VID \neq V2.VIDandV1.EID=V2.EID} (V2))$

### 3.10 Query 10: Consider all of the countries that have won at least one event for these countries name the first and last name of its players who achieved the highest number of wins

First we find the Player with the highest number of wins and we cross join that with the players Country

$P1 := \sigma_{wins>0}(Player)$
$P2 := \sigma_{wins>0}(Player)$

$P4 := \Pi_{P1.PID,P1.fname,P1.lname.P1.gender,P1.activeSince,P1.wins,P1.losses,P1.totalPoints,P1.globalRank,P1.ctryID}$
$((P2) \bowtie_{P2.wins>P1.wins} (P1))$

$P3 := Player - P4$
$Pi_{name,fname,lname}(P3) \bowtie_{ctryID=CTRYID} (Country)$