Whitney Kenner
U0777962
6/6/23

Part 1 - Joins

Consider the following relations:

1. $T1 \bowtie_{T1.A=T2.A} T2$
2. $T1 \bowtie_{T1.Q=T2.B} T2$
3. $T1 \bowtie T2$
4. $T1 \bowtie_{T1.A=T2.A \wedge T1.R=T2.C} T2$

1.

T3					
A_{t1}	Q	R	A_{t2}	B	C
20	a	5	20	b	6
20	a	5	20	b	5

2.

T4					
A_{t1}	Q	R	A_{t2}	B	C
25	b	8	20	b	6
25	b	8	20	b	5

3.

T5				
A	Q	R	B	C
20	a	5	b	6
20	a	5	b	5

4.

T6					
A_{t1}	Q	R	A_{t2}	B	C

T6					
20	a	5	20	b	5

Part 2 - Chess Queries

Consider the schemas below for a Chess database, similar to the one you designed in Lab 2, except some columns are left out for simplicity.

The "eID" column in Events is a simplification of the primary key and represents a unique Event. The "eID" in Games is a foreign key referencing eID from Events.

"pID" in Players is a simplification of the primary key and represents a unique player. The Name column in Players is still required to be unique, as an additional candidate key.

"wpID" and "bpID" in Games represent white players and black players, respectively, and they are foreign keys referencing pID from Players.

Hint: you can't natural join Games with Players because they have no columns with the same name.

Note that the instance data given is just to show an example, and it may change (you can't hard-code your queries based on these instances).

Recall that in this database, a result of "1-0" means white won, "0-1" means black won, and "1/2-1/2" means it was a draw.

Events

Name	Year	eID
World Championship	1987	1
Moscow Open	2018	2
World Championship	2018	3

Players

Name	Elo	pID
Magnus Carlsen	2882	1
Judit Polgar	2735	2
Fabiano Caruana	2844	3
Gary Kasparov	2851	4
Anatoly Karpov	2780	5

Games

gID	eID	Result	wpID	bpID
1	3	1/2-1/2	1	3

2	3	1/2-1/2	3	1
3	2	1-0	2	1
4	1	1/2-1/2	4	5
5	3	0-1	3	1

Write relational algebra queries for the following. You can (and should) write your query on multiple lines if you use the renaming operator.

1. Find the names of any player with an Elo rating of 2850 or higher.

   $\pi_{\text{Name}}(\sigma_{\text{ELO} > 2850}(\text{Players}))$

2. Find the names of any player who has ever played a game as white.

   $\pi_{\text{Name}}(\text{Games} \bowtie_{\text{Gmaes.wpId} = \text{Players.pID}} \text{Players})$

3. Find the names of any player who has ever won a game as white.

   $\pi_{\text{Name}}(\text{Players} \bowtie_{\text{Games.wpId} == \text{Players.pID}} (\sigma_{\text{Result} = \text{“1-0”}}(\text{Games})))$

4. Find the names of any player who played any games in 2018.

   $\pi_{\text{Name}}(\text{Players} \bowtie_{\text{Players.pId} == \text{Games.wpID} \vee \text{Players.pID} == \text{Games.bpID}} (\text{Games} \bowtie (\sigma_{\text{year} = 2018}(\text{Events}))))$

5. Find the names and dates of any event in which Magnus Carlsen lost a game.

   $\pi_{\text{Name, year}}(\text{Events} \bowtie (\text{Games} \bowtie_{(\text{Players.pID} == \text{Games.wpID} \wedge \text{Games.Result} == \text{“1-0”}) \vee (\text{Players.pID} == \text{Games.bpID} \wedge \text{Games.Result} == \text{“0-1”})} (\sigma_{\text{name} = \text{“Magnus Carlsen”}} (\text{Players}))))$

6. Find the names of all opponents of Magnus Carlsen. An opponent is someone who he has played a game against. Hint: Both Magnus and his opponents could play as white or black.

   P (MagnusTable, $\sigma_{\text{name} = \text{“Magnus Carlsen”}} (\text{Players})$)

   $\pi_{\text{Name}}(\sigma_{\text{name} \neg \text{“Magnus Carlsen”}} (\text{Games} \bowtie_{\text{Games.wpID==Magnus.pID} \vee \text{Games.bpID} == \text{Magnus.pID}} \text{MagnusTable}))$

Part 3 - LMS Queries

Part 3.1:
a) Provide the relation that is the result of the following query. Your relation should be in the form of a table, and should include the schema.

Name
Hermoine
Harry

b) Provide a simple English description of what the query is searching for. Your description should be in general terms (remember that the original LMS instance data may change).

- The first part is creating a table of the student IDs that have received a C grade (this relation is called C). The second part is taking all of the student IDs in the relation "enrolled" and subtracting the relation C from it, which leaves us with all student IDs of those who have not received Cs. then the names are projected from this relation which gives us Harry and Hermoine

Part 3.2:
a) Provide the relation that is the result of the following query. Your relation should be in the form of a table, and should include the schema.

Name
Hermoine

b) Provide a simple English description of what the query is searching for. Your description should be in general terms (remember that the original LMS instance data may change).

- First we are creating 2 copies of Students and calling them s1 and s2 respectively. Then we take the cross product (multiply the tables by each other). Then we select from this table based on a few conditions including that s1 name is Ron, the s1 and s2 birthday are the same and s2 name is not the same as s1 name (Ron). this leaves us with 1 row, from this 1 row table we pull out the s2 name which gives us Hermoine 🙂

Part 3.3:
a) Provide the relation that is the result of the following query. Your relation should be in the form of a table, and should include the schema.

(empty table)

b) Provide a simple English description of what the query is searching for. Your description should be in general terms (remember that the original LMS instance data may change).

- We begin by pulling out the cID and sID from enrolled and pulling out the sID from students, then we go through a lengthy division process that results in an empty table. From there we natural join courses with this empty temporary relation BUT for natural join to work we need to have 1 column from each table with the same name, so they cannot be natural joined, this means when we try to pull out the cnames from this relation, it is also an empty relation and nothing is returned

Part 4

Provide a relational algebra query that uses the divide operator to find the names of all students who are taking all of the 3xxx-level classes.

$P(\text{Level3Courses}, \sigma_{cID > 3000 \wedge cID < 4000}(\text{Courses}))$

$\pi_{Name}(\pi_{sID, cID}(\text{Enrolled}) / \pi_{cID}(\text{Level3Courses}))$