

CS 178: Homework 2 Solutions

The database you will use for the first three problems of this assignment contains information about people, vacation locations, and when those people visited them. The relations are:

- Person(ID, Name)
- Location(PlaceName, Country, MainAttraction)
- Visited(ID, PlaceName, Year).

You should assume that place names are unique in the world.

Ques.1: Consider the schema given above. (a) Give a primary key for each relation. Are there any relations for which there is an alternate candidate key which you have not chosen as the primary key? Why or why not? (b) State all referential integrity constraints (inclusion dependencies) that should hold on these relations.

Solution: (a) Person(ID), Location(PlaceName), and Visited(ID,PlaceName,Year), depending on whether you have one record each year indicating that a person visited it then, or one for each time they visited. There are no alternate keys, except for Person(Name) if you assume that to be unique. People might argue that Location(PlaceName,Country) is a key, since place names are duplicated in different countries.

(b) Visited(ID) references Person(ID), and Visited(PlaceName) references Location(PlaceName).

Ques.2: Write relational algebra queries for each of the queries below. In addition, write tuple relational calculus queries for at least the first three queries. If a query is long, feel free to break it up into a series of queries with intermediate answers stored using the assignment operator in temporary relations (e.g. $r \leftarrow x$). Note: Your queries must not be dependent on the instance of the database (i.e., they should work without modification even if another instance of the database is given) and ignore issues of duplicates.

- Return all tuples in Location where the main attraction is a beach.
 - Solution: $\sigma_{Mainattraction='Beach'}(Location)$ is the RA query.
 - In tuple calculus the query is $\{L \mid L \in Location \wedge L.Mainattraction = Beach\}$
- Return all the places visited by someone before 2000.
 - Solution: $\pi_{PlaceName}\sigma_{Year < 2000}(Visited)$
 - In tuple calculus: $\{p \mid V \in Visited \wedge V.Year < 2000 \wedge V.PlaceName = p\}$
- Return the names of everybody who visited somewhere in 2003.
 - Solution: $\pi_{Name}\sigma_{Year=2003}(Visited \bowtie Person)$ where \bowtie is the natural join between Visited and Person, i.e., $\sigma_{Visited.ID=Person.ID}(Visited \times Person)$.
 - In tuple calculus, $\{n \mid P \in Person \wedge V \in Visited \wedge P.ID = V.ID \wedge V.year = 2003 \wedge P.Name = n\}$
- Return pairs of names of people that visited the same country in the same year. The result should have schema (Name1, Name2).

- Solution: This query requires using the rename operator ρ . Be careful about using natural joins – you will have to rename attribute names to use natural join. Below is a query that uses the assignment operator, but we can write it without this also by simply substituting the expression for $temp$ into the reference for $temp$ in the second line.

$$temp \leftarrow \sigma_{(V1.ID \neq V2.ID) \wedge (V1.PlaceName = V2.PlaceName) \wedge (V1.Year = V2.Year)}((\rho_{V1}(Visited)) \times (\rho_{V2}(Visited)))$$

$$\pi_{P1.name, P2.name}(\sigma_{(P1.ID = V1.ID) \wedge (P2.ID = V2.ID)}(temp \times (\rho_{P1}(Person)) \times (\rho_{P2}(Person))))$$

- In tuple calculus: $\{P | P_1 \in Person \wedge P_2 \in Person \wedge (P.ID_1 = P_1.ID) \wedge (P.ID_2 = P_2.ID) \wedge (P_1.ID \neq P_2.ID) \wedge (V_1 \in Visited) \wedge (V_2 \in Visited) \wedge (V_1.Country = V_2.Country) \wedge (V_1.Year = V_2.year) \wedge (P_1.ID = V_1.ID) \wedge (P_2.ID = V_2.ID)\}$

- Return the names of the unfortunate people who didnt visit anywhere at all.

- Solution: $\pi_{Name}(Person \bowtie (\pi_{ID}(Person) - \pi_{ID}(Visited)))$ where \bowtie is the natural join on attribute ID.

- In tuple calculus we have: $\{n | P \in Person \wedge (P.Name = n) \wedge (\nexists V \in Visited (V.ID = P.ID))\}$

- Return the names of the lucky people who visited all of the places listed. You may assume that every PlaceName in Location occurs in Visited.

- Solution: This is where the division operator is useful –read about this in the textbook. $\pi_{Name} Person \bowtie (\pi_{ID, PlaceName}(Visited \div (\pi_{PlaceName} Location)))$

- In tuple calculus we have: $\{n | P \in Person \wedge (P.Name = n) \wedge (\forall L \in Location (\exists V \in Visited (L.PlaceName = V.PlaceName \wedge V.ID = P.ID)))\}$

Ques.3: For the aforementioned schema, what does the following query compute: $\pi_{PlaceName}(Location) - \pi_{PlaceName}(\sigma_{Year > 2003}(Visited))$.

Solution: The places that were not visited by anyone in or after 2003.

For the next two questions (Questions 4,5) you will use a database that stores information on employees and the departments and projects they work for.

Ques.4: Write relational algebra queries for the following:

- Find the names and addresses (street and city) of employees who work for the Research department.

- To get name and address of employee we need to access EMP table; to get name of department we need to access DEPT table. The join condition between the two relations is on the department number attribute. Select tuples where the department name is equal to Research.

– Solution: $\pi_{Name,Street,City}(\sigma_{(EMP.DNO=DEPT.DNUM)\wedge(Dname='Research')}(EMP \times DEPT))$

- For every project located in Stafford, list the project number, the controlling department number and the manager's name.

– Solution: We need to access PROJECTS to figure out project information such as the project location and the controlling department. Once we have the controlling department, we need to look up the DEPT relation to figure out the Manager's social security number. We then need to look up EMP to find the name of the manager. The join conditions between PROJECTS and DEPT are on department number, and the join condition between DEPT and EMP is on SSN.

$$\pi_{EMP.Name,PROJECTS.PNum,DEPT.DNUM}(\sigma_{(EMP.SSN=DEPT.MGRSSN)\wedge(DEPT.DNUM=PROJECTS.DNO)\wedge(PROJECTS.PLOCATION='Stafford')}(EMP \times (DEPT \times PROJECTS)))$$

- For each employee, find their name and their supervisor's name.

– Solution: To find the supervisor's name we need to access EMP again. Therefore we need a query where the relation EMP is queried twice – suggesting that we need a rename operator. Rename EMP to EMP2 – this copy will hold information about the supervisor's name. Join EMP and EMP2 based on supervisor SSN.

$$\pi_{EMP.Name,EMP2.Name}(\sigma_{(EMP.SuperSSN=EMP2.SSN)}(EMP \times (\rho_{EMP2}(EMP))))$$

Ques.5: For each of the relational algebra queries below, describe what the query computes.

(a):

$$\pi_{SSN}(EMP) - \pi_{SuperSSN}(EMP)$$

This query finds all employees who are not supervisors. The first project lists all employees in the Company, and the second lists all supervisors in the Company.

(b)

$$\pi_{W2.ESSN}(\sigma_{(WORKS.pno=W2.pno)}(\pi_{WORKS.pno}(\sigma_{(name='Smith')\wedge(ESSN=SSN)}(WORKS \times EMP)) \times (\rho_{W2}(WORKS))))$$

Solution: Consider the third line of the query. The join condition is on SSN of an employee. The tuples in the result associate all the employee information with the employee's SSN in the

WORKS relation. The selection condition Name='Smith' then selects only tuples associated with an employee named Smith. Thus, this select and join finds all tuples associated with an employee named Smith and projects out all the projects (numbers) that Smith works on. The fourth line of the query essentially creates a copy of the WORKS relation with the name W2. The second line is a predicate that forms a join of result of line 3 (which is a relation containing project numbers where Smith works) with W2 where the two tuples have the same project number. In other words, tuples in W2 associated with the same projects that Smith works on are selected in the result. Finally, the SSNs of the employees who work on the same projects as Smith are projected in the result.