Homework 2 Databases

Jan C. Bierowiec

Question 1

The relational algebra is closed under the operators. Explain what this means and give an example.

In relational algebra, the concept of "closure" means that the result of any operation is always a relation itself. This is why relational algebra is closed under operators. In relational algebra, all operations take relations as an input and then produce a new relation as an output. In order for this do be done correctly, a close environment needs to be instilled for the logic to perform correctly and for the user to extract the relations they desire. This property also allows relational algebra expressions to be composed and nested, as the output of one operation can be used as the input to another. This then allows for more complex operations which can be performed on large datasets and can provide valuable information with respect to specifying certain attributes and noticing trends as a result.

An example of relational algebra being closed under the operators is if you take an employee relation, such as employee(name, city, salary) and a department relation, such as department(name, manager), and you would like to know the number of employees that live in a city like New York City for example. To obtain this operation, we first need to use the selection operator σ and specify the city we are referring to, which in this case is New York City. The set up would thus look like the following:

$$\sigma_{\text{city='New York City'}}(employee)$$

The result of using this operator in a closed environment will be a relation that consists of all the employees who strictly live in New York City. We can then apply another operator, a projection operator Π in order to extract the names of the employees who live in New York City. The set up for this operation would look like the following:

$$\Pi_{\text{name}}(\sigma_{\text{city='New York City'}}(employee))$$

The result of this new operation outputs the names of all the employees that live in New York City. It can therefore be shown how one relation can be created under closed operators to create a new relation, in this case the relation being all the employees in New York City, and then we can create another relation using that previously calculated relation to create a new relation that contains all the names of the employees who live in New York City. All of this of course was done with operators under a closed environment. Therefore each operation results in a relation, which demonstrates closure.

Question 2

SQL is a declarative data manipulation language. What are some pros and cons of declarative DMLs relative to procedural ones?

Some pros of declarative DMLs compared to procedural DMLs are how declarative DMLs have an ease of use, meaning languages like SQL allow a user to focus on the "what" they want to know and not necessarily the "how" to obtain it, which makes it more user-friendly and easy to use. This means that relational calculus is performed. In relation to the previous homework the DBMS makes the execution of queries of a declarative DML like SQL to be optimized. This means that the users do not need to manually specify the most efficient way to fetch the data. SQL is also better as a declarative DML because there is less code to write, than in the case of a procedural DML.

Some cons of declarative DMLs compared to procedural DMLs are how declarative DMLs have less control, making the execution details more abstract. This is a disadvantage especially when higher or quicker performance is needed. In relation to question 1, another disadvantage is how complex queries can become. In the case of recursion or complex join operations, queries can become very long and can be difficult to express. This is most commonly because in procedural DMLs relational algebra is performed. Lastly, declarative DMLs have a learning curve if someone has a procedural background based on how unintuitive declarative DMLs can be. In procedural languages on the other hand, the query specifies the (high-level) strategy to find the desired result based on sets.

Question 3

Consider the following expressions, which use the result of a relational algebra operation as the input to another operation. For each expression, explain in words what the expression does.

1. $\sigma_{year \geq 2009}(takes) \bowtie student$

2. $\sigma_{year \geq 2009}(takes \bowtie student)$
3. $\Pi_{ID,name,course\_id}(student \bowtie takes)$

$$\sigma_{year \geq 2009}(takes) \bowtie student$$

The expression above first performs a selection (σ) operation on the relation *takes* with the filer being *year ≥ 2009*. After this selection operation is completed, then the resulting relation is then used as an input to perform a natural join operation ($\bowtie$) with the *student* relation. The natural join operation will then create a new relation of students that took a course in 2009 or later.

$$\sigma_{year \geq 2009}(takes \bowtie student)$$

The expression above first performs a natural join ($\bowtie$) operation on the *takes* and *student* relations which creates a new table of these combined relations. After this is completed, a selection (σ) operation is performed with the filer being *year ≥ 2009*. This is applied to the previous joined relation result. The output of this query is the same as in the previous case however the order is different. In this case, the expression first joins the *takes* and *student* relations and from there they are filtered based on the year selection of courses in 2009 or later.

$$\Pi_{ID,name,course\_id}(student \bowtie takes)$$

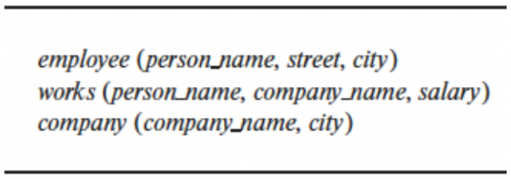
The expression above similarly as in the previous case first performs a natural join ($\bowtie$) operation on the *student* and *takes* relations (with the order flipped) which creates a new table of these combined relations. Then a projection (Π) operation takes place which selects only the *ID*, *name*, and *course_id* attributes from the previously resulted joined table. This expression therefore returns a list of student IDs, the corresponding student names, and the course ID's of the courses the students are enrolled in when the *student* and *takes* relations were previously performed.

Question 4

Consider the relational database of Figure 1. Give an expression in the relational algebra to express each of the following queries:

1. Find the names of all employees who live in city "Miami".

2. Find the names of all employees whose salary is greater than \$100,000.
3. Find the names of all employees who live in “Miami” and whose salary is greater than \$100,000.



employee (*person_name*, *street*, *city*)
works (*person_name*, *company_name*, *salary*)
company (*company_name*, *city*)

Fig 1 Employee Database

Find the names of all employees who live in city “Miami”.

For the first query, we would look only at the *employee* relation. We would only consider this relation because the query requests the names of all the employees who live in the city Miami. Both *city* and *person_name* are attributes of the *employee* relation, so the expression in relational algebra would look like the following:

$$\Pi_{person\_name}(\sigma_{city='Miami'}(employee))$$

This expression would first select using the selection (σ) operator from the *employee* relation the *city* attribute, specifying it to be “Miami”. Then once the employees who live in Miami are narrowed down, the projection (Π) function is used to extract and project the names of all the employees who live in Miami.

Find the names of all employees whose salary is greater than \$100,000.

For the second query, we would look only at the *works* relation. We would only consider this relation because the query requests the names of all the employees whose salary is greater than \$100,000. Both *salary* and *person_name* are attributes of the *works* relation, so the expression in relational algebra would look like the following:

$$\Pi_{person\_name}(\sigma_{salary>100000}(works))$$

Find the names of all employees who live in "Miami" and whose salary is greater than $100,000.

This expression would first select using the selection (σ) operator from the *works* relation the *salary* attribute, specifying it to be "100000". Then once the employees whose salary is greater than $100,000 are narrowed down, the projection (Π) function is used to extract and project the names of all the employees whose salary is greater than $100,000.

For the third query, we would look only at both the *person_name* and *works* relation. We would only consider these relations because the query requests the names of all the employees who live in the city Miami and whose salary is greater than $100,000. Both *salary* and *person_name* are attributes of the *works* relation and both *city* and *person_name* are attributes of the *employee* relation, so the expression in relational algebra would look like the following:

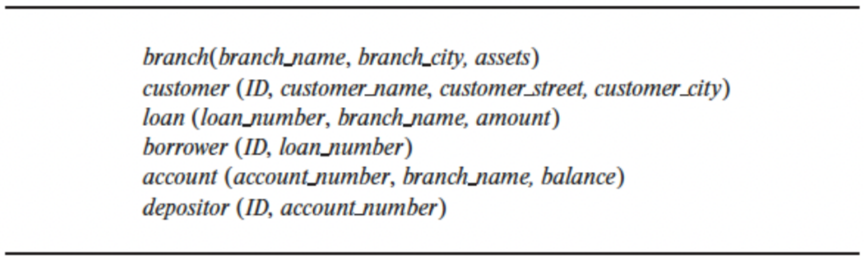
$$\Pi_{person\_name}(\sigma_{city='Miami'} \wedge salary > 100000(employee \bowtie_{person\_name} works))$$

This expression would first do a natural join operation using the join ($\bowtie$) operator between the *employee* and *works* relations on the common attribute they share, that being *person_name.* Then once a new relation is created of person names of employees that work, a selection operation is performed using the selection (σ) operator where the attribute of *city* from the relation *employee* is specified to be 'Miami' and that is compared with an and ($\wedge$) operation to the *salary* attribute, from the relation *works.* Then when a *person_name* from both the *employee* and *works* attributes are joined, and the attribute of *city* and *salary* are specified to be 'Miami' and 100000, then finally the *person_name* can be projected using the projection (Π) function. This will yield a new relation of all the employee names who both work in Miami and whose salary is greater than $100,000.

Question 5

Consider the bank database of Figure 2. Give an expression in the relational algebra for each of the following queries:

1. Find the names of all branches located in "Chicago".
2. Find the names of all borrowers who have a loan in branch "Down-town".



branch(*branch_name, branch_city, assets*)
customer (*ID, customer_name, customer_street, customer_city*)
loan (*loan_number, branch_name, amount*)
borrower (*ID, loan_number*)
account (*account_number, branch_name, balance*)
depositor (*ID, account_number*)

Fig 2 Bank Database

Find the names of all branches located in "Chicago".

To find the names of all the branches located in "Chicago", we would need to look for all the branch_names and we would need to specify that the city we are looking for the branch names is "Chicago. The relational algebra formula based on the figure would thus look like the following:

$$\Pi_{\text{branch\_name}}(\sigma_{\text{branch\_city='Chicago'}}(branch))$$

What this relational algebra formula does, is it first selected from the *branch* tuples the *branch_city* attribute, specifying it to be Chicago, and once we narrow down all the branches from Chicago, we can then extract the name of the branch by using the project function to project all the branch names of the branches that are located in Chicago.

Find the names of all borrowers who have a loan in branch "Down-town".

To find the names of all the borrowers who have a loan in the branch "Down-town", we would need to look for all customers who are borrowers and who have a loan under the branch_name of "Down-town". The relational algebra formula based on the figure would thus look like the following:

$$\Pi_{\text{customer\_name}}(\sigma_{\text{branch\_name='Down-town'}}((borrower \bowtie loan \bowtie branch) \bowtie customer))$$

What this relational algebra formula does, is it first joins a borrower to a loan as well as to the branch from where they borrowed from. From there we need to join this information to a customer. Then once we narrow down customers who are borrowers that have a loan and belong to a branch, we can then select the branch to be “Down-town”. From here, the table gets narrowed down to a customer who is a borrower, has a loan, and belongs to the “Down-town” branch and from here we can project the customer names to be able to identify each borrower.

we would need to look for all the branch_names and we would need to specify that the city we are looking for the branch names is “Chicago. The relational algebra formula based on the figure would thus look like the following: