CSC 370

Activity Worksheet:
Expressing Constraints in Relational Algebra

Dr. Sean Chester

Fall 2022

Notes

This worksheet should provide extra practice questions for writing relational constraints using relational algebra beyond the set of questions in the textbook that we have already reviewed together.

Questions

All seven constraints should be upheld on a relational schema with the following structure:

BankSystem Schema:

Member(member_id, social_insurance_number, date_joined)
Account(account_id, member_id, balance, credit_limit)
Transactions(transaction_id, send_account_id, recipient_account_id, date, amount)

1. Non-Negative Balances

   Constraint: No account can have a balance below zero.

   Solution:

2. Within Credit Limits

   Constraint: No account can have a negative balance that exceeds the credit limit.

   Solution:

3. CRA Agrees with Your Member Count

   Constraint: No two members can have the same social insurance number.

   Solution:

4. No Dangling Members

   Constraint: Every member must have at least one account.

   Solution:

5. No Transfers Within an Account

   Constraint: Every transaction has a unique sender and recipient account.

   Solution:

6. No Self-Transfers

   Constraint: Every transaction is between a unique sending member id and recipient member id.

   Solution:

7. Junior Account Limits

   Constraint: No member who joined within the past two years can have an account with a credit limit over $5000.

   Solution:

8. Daily Limit

   Constraint: Each account is limited to three outgoing transactions per day.

   (Note: this might sound very artificial, but that is because we have not yet learned about aggregation functions so we cannot create constraints on the sum of the transactions in a day.)

   Solution:

Solutions

Question 1

$$\sigma_{\text{balance}<0}(\text{Account}) = \emptyset$$

To enforce the constraint, we want to ensure that no records can violate it; i.e., if we retrieve all records that meet the negation of the constraint, we will always get the empty set.

Here, we retrieve all accounts with a negative balance using the selection operator.

Question 2

$$\sigma_{\text{balance}>\text{credit\_limit} * -1}(\text{Account}) = \emptyset$$

This question was a bit confusing in that it compares a negative balance to a credit limit and requires some arithmetic inside the constraint (not to be expected on the exam). The key difference to the question above is that we compare two attributes within a relation rather than comparing one attribute to a constant literal.

Question 3

$$\rho_A(\text{Member}) \bowtie_{\text{A.member\_id}\neq\text{B.member\_id AND A.s\_i\_n=B.s\_i\_n}} \rho_B(\text{Member}) = \emptyset$$

To compare two tuples from the same relation, we use a self-join, i.e., a join operation in which the same table appears as both left-hand and right-hand operands. To disambiguate columns in the join predicate, we use rho to rename the relations.

In this case, because member_id is a key to the relation, we can use that to ensure that the two tuples that we are comparing are not just the same tuple taken once from each copy of the relation.

Question 4

$$\pi_{\text{member\_id}}(\text{Member}) \setminus \pi_{\text{member\_id}}(\text{Account}) = \emptyset$$

If every member has at least one account, then every member id must appear in the projection of Account on the member_id field. We can use the set difference operator to ensure there are no member ids in the Member relation that do not appear in the Account relation.

Question 5

$$\sigma_{\text{sender\_account\_id = recipient\_account\_id}}(\text{Transaction}) = \emptyset$$

This question is just a warm-up for the next one. It is effectively the same as Question 2.

Question 6

$$\rho_{A1}(\text{Account}) \bowtie_{\text{A1.member\_id = A2.member\_id}}$$

$$\rho_{A2}(\text{Account}) \bowtie_{\text{A1.account\_id = sender\_account\_id AND A2.account\_id = recipient\_account\_id}} \text{Transactions} = \emptyset$$

Observe that because these relations have been normalised, the member id information is not stored with the transactions. Thus, we need to join each account id in the transaction to the Account relation so that we can retrieve the member id information.

Question 7

$$\sigma_{\text{data\_joined}>\text{'2019-09-28'}}(\text{Member}) \bowtie \sigma_{\text{credit\_limit}>5000}(\text{Account}) = \emptyset$$

As in the previous question, we need to join together (this time with a natural join) the two normalised tables so that we can link information about the credit limit to the date that a member joined the institution.

Question 8

$$\sigma_{\text{T1.transaction\_id} \neq \text{T2.transaction\_id} \neq \text{T3.transaction\_id} \neq \text{T4.transaction\_id}} \Big(\rho_{\text{T1}}(\text{Transactions}) \bowtie_{\text{T1.date} = \text{T2.date AND T1.sender\_account\_id} = \text{T2.sender\_account\_id}} \rho_{\text{T2}}(\text{Transactions}) \bowtie_{\text{T1.date} = \text{T3.date AND T1.sender\_account\_id} = \text{T3.sender\_account\_id}} \rho_{\text{T3}}(\text{Transactions}) \bowtie_{\text{T1.date} = \text{T4.date AND T1.sender\_account\_id} = \text{T4.sender\_account\_id}} \rho_{\text{T4}}(\text{Transactions}) \Big) = \emptyset$$

The negation of this constraint is that we can find four transactions on the same day from the same sender account id. We self-join a relation three times to find a set of four rows in that relation instance to examine and set their transaction ids to all be distinct to ensure that we have four different rows (since transaction id is a key for the relation). This machinery allows us to compare four transactions to each other.

To test the constraint, we set the join predicate to capture its negation, i.e., that all four transactions have the same sender account and date.