Tutorial: Calculus with a pinch of Algebra

This tutorial uses the schema of the database in Tutorial 1.

1. Translate the following queries into tuple relational calculus (TRC).

   (a) Find the different departments in School of Computing.

   Solution:
   $\{T \mid \exists T_1\ (T_1 \in department \wedge T_1.faculty = \text{'School of Computing'} \wedge T.department = T_1.department)\}$
   Although the following notations and combinations thereof exist. They are not used in this module. They are simpler, but may hide some difficulties (related to quantification) that we want to make explicit.
   $\{T \mid \exists T_1 \in department\ (T_1.faculty = \text{'School of Computing'} \wedge T.department = T_1.department)\}$
   $\{< T_1.department > \mid \exists T_1\ (T_1 \in department \wedge T_1.faculty = \text{'School of Computing'})\}$
   Note that Calculus (and algebra), as we study them, work on sets. Therefore they automatically eliminate duplicates.

   (b) Let us check the integrity of the data. Find the emails of the students who borrowed or lent a copy of a book before they joined the university. There should not be any.

   Solution: $\{T \mid \exists T_1\ \exists T_2\ (T_1 \in student \wedge T_2 \in loan \wedge ((T_1.email = T_2.borrower \wedge T_2.borrowed < T_1.year) \vee (T_1.email = T_2.owner \wedge T_2.borrowed < T_1.year)) \wedge T.email = T_1.email)\}$
   This can be rewritten as follows.
   $\{T \mid \exists T_1\ \exists T_2\ (T_1 \in student \wedge T_2 \in loan \wedge (T_1.email = T_2.borrower \vee T_1.email = T_2.owner) \wedge T_2.borrowed < T_1.year \wedge T.email = T_1.email)\}$
   We can accordingly write the answer in Tutorial 2 as either:

```
SELECT DISTINCT s.email
FROM loan l, student s
WHERE (s.email = l.borrower AND l.borrowed < s.year)
OR (s.email = l.owner AND l.borrowed < s.year);
```

```
SELECT DISTINCT s.email
FROM loan l, student s
WHERE (s.email = l.borrower OR s.email = l.owner)
AND l.borrowed < s.year;
```

   (c) Print the emails of the students who borrowed but did not lend a copy of a book on the day that they joined the university.

Solution: $\{T \mid \exists T_1\ \exists T_2\ (T_1 \in student \wedge T_2 \in loan\ \wedge\ T_1.email = T_2.borrower\ \wedge\ T_2.borrowed = T_1.year\ \wedge \neg(\exists T_3\ (T_3 \in loan\ \wedge\ T_1.email = T_3.owner\ \wedge\ T_3.borrowed = T_1.year))\ \wedge\ T.email = T_1.email)\}$

This translates into SQL as follows.

```
SELECT  DISTINCT s.email
FROM loan l2, student s1
WHERE s1.email = l2.borrower AND l2.borrowed = s1.year
AND NOT EXISTS(
    SELECT  *
    FROM loan l3
    WHERE s.email = l3.owner AND l3.borrowed = s1.year;
```

The query can also be written as follows in calculus.

$\{T \mid \exists T_1\ \exists T_2\ \forall T3\ (T_1 \in student \wedge T_2 \in loan\ \wedge\ T_1.email = T_2.borrower\ \wedge\ T_2.borrowed = T_1.year \wedge \neg(T_3 \in loan \wedge T_1.email = T_3.owner \wedge T_3.borrowed = T_1.year) \wedge T.email = T_1.email)\}$

$\{T \mid \exists T_1\ \exists T_2\ \forall T3\ (T_1 \in student \wedge T_2 \in loan\ \wedge\ T_1.email = T_2.borrower\ \wedge\ T_2.borrowed = T_1.year \wedge (T_3 \notin loan \vee T_1.email \neq T_3.owner \vee T_3.borrowed \neq T_1.year) \wedge T.email = T_1.email)\}$

(d) Find the ISBN13 of the books that have been borrowed by all the students in the computer science department.

Solution: $\{T \mid \exists T_1\ (T_1 \in book \wedge \forall T_2\ ((T_2 \in student \wedge T_2.department = \text{'Computer Science'}) \rightarrow (\exists T_3\ (T_3 \in loan\ \wedge\ T_2.email = T_3.borrower\ \wedge T_1.isbn13 = T_3.book))\ \wedge T.isbn13 = T_1.isbn13)\}$

Which can be rewritten as follows.

$\{T \mid \exists T_1\ (T_1 \in book \wedge \neg(\exists T_2\ (T_2 \in student \wedge T_2.department = \text{'Computer Science'} \wedge \neg(\exists T_3\ (T_3 \in loan\ \wedge\ T_2.email = T_3.borrower\ \wedge T_1.isbn13 = T_3.book))))\ \wedge T.isbn13 = T_1.isbn13)\}$

This translates into SQL as follows.

```
SELECT   b.isbn13
FROM book b
WHERE NOT EXISTS(
    SELECT  *
    FROM student s
    WHERE s.department = 'Computer Science'
        AND NOT EXISTS(
        SELECT  *
        FROM loan l
        WHERE s.email = l.borrower
        AND b.isbn13 = l.book));
```

2. Translate the following queries into relational algebra.

(a) Find the different departments in School of Computing.

Solution:

$\pi_{d.department}(\sigma_{d.faculty=\text{'School of Computing'}}(\rho(department, d)))$

(b) Let us check the integrity of the data. Find the emails of the students who borrowed or lent a copy of a book before they joined the university. There should not be any.

Solution: $\pi_{s.email}(\sigma_{(s.email=l.borrower\ \vee\ s.email=l.owner)\ \wedge\ l.borrowed<s.year}(\rho(student, s) \times \rho(loan, l)))$

This can also be written with a join.

$\pi_{s.email}(\rho(student, s) \bowtie_{(s.email=l.borrower\ \vee\ s.email=l.owner)\ \wedge\ l.borrowed<s.year} \rho(loan, l))$

or as

$\pi_{s_1.email}(\sigma_{s_1.email=l_1.borrower\ \wedge\ l_1.borrowed<s_1.year}(\rho(student, s_1) \times \rho(loan, l_1)))$

$\cup$

$\pi_{s_2.email}(\sigma_{s_2.email=l_2.owner\ \wedge\ l_2.borrowed<s_2.year}(\rho(student, s_2) \times \rho(loan, l_2)))$

We can accordingly write the answer in Tutorial 2 as either:

```
SELECT  s1.email
FROM loan l1, student s1
WHERE s1.email = l1.borrower
AND l1.borrowed < s1.year
UNION
SELECT  s2.email
FROM loan l2, student s2
WHERE s2.email = l2.owner
AND l2.borrowed < s2.year
```

(c) Print the emails of the students who borrowed but did not lend a copy of a book on the day that they joined the university.

Solution: We use non-symmetric set difference.

$\pi_{s_1.email}(\sigma_{s_1.email=l_1.borrower \wedge l_1.borrowed=s_1.year}(\rho(student, s_1) \times \rho(loan, l_1)))$

$\setminus$

$\pi_{s_2.email}(\sigma_{s_2.email=l_2.owner \wedge l_2.borrowed=s_2.year}(\rho(student, s_2) \times \rho(loan, l_2)))$

This is the query with `EXCEPT` in tutorial 2.

Non-symmetric set difference can be used to implement other universally quantified queries in algebra but these queries are quite complicated to write.

References

[1] S. Bressan and B. Catania. *Introduction to Database Systems.* McGraw-Hill Education, 2006.

[2] R. Elmasri and S. B. Navathe. *Fundamentals of Database Systems.* Pearson, 7th edition, 2015.

[3] R. Ramakrishnan and J. Gehrke. *Database Management Systems.* McGraw-Hill, 2002.