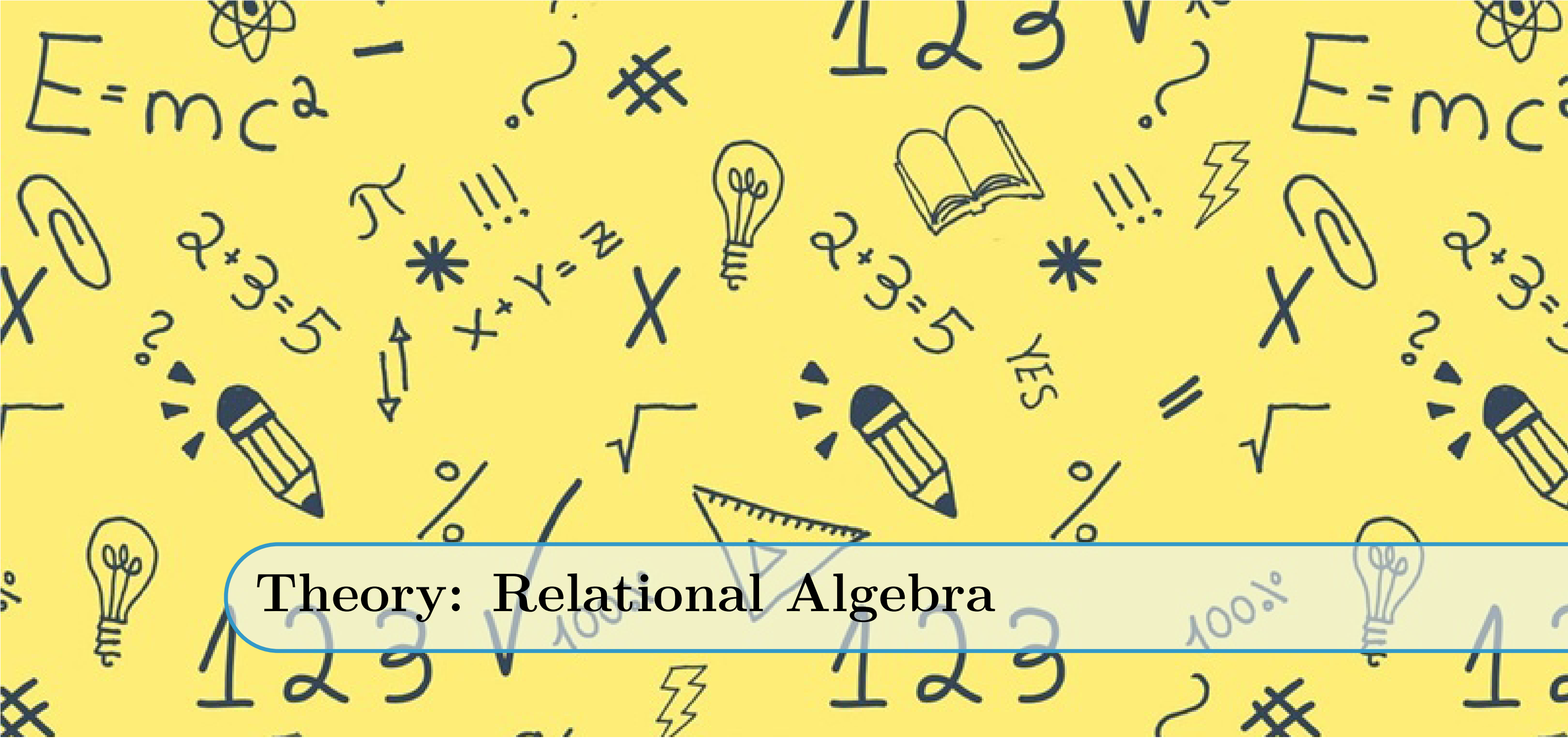

Theory: Relational Algebra

Students at the National University of Ngendipura (NUN) buy books for their studies. They also lend and borrow books to and from other students. Your company, Apasaja Private Limited, is commissioned by NUN Students Association (NUNStA) to implement an online book exchange system that records information about students, books that they own and books that they lend and borrow.

The database records the name, faculty, and department of each student. Each student is identified in the system by her email. The database also records the date at which the student joined the university (year attribute).

The database records the title, authors, publisher, year and edition and the ISBN-10 and ISBN-13 for each book. The International Standard Book Number, ISBN-10 or -13, is an industry standard for the unique identification of books. It is possible that the database records books that are not owned by any students (because the owners of a copy graduated or because the book was advised by a lecturer for a course but not yet purchased by any student.)

The database records the date at which a book copy is borrowed and the date at which it is returned. We refer to this information as a loan record.

For auditing purposes the database records information about the books, the copies and the owners of the copies as long as the owners are students or as there are loan records concerning the copies. For auditing purposes the database records information about graduated students as long as there are loan records concerning books that they owned.

This tutorial uses the schema from "SQL: Creating and Populating Tables" including all the updates done during the tutorial.

Questions

Not all questions will be discussed during tutorial. You are expected to attempt them before coming to the tutorial. You may be randomly called to present your answer during tutorial. You are encouraged to discuss them on Canvas Discussion.

1. **Relational Algebra.**

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

Comments:

One possible relational algebra expression.

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

This corresponds to the following SQL query.

Code: Query

```sql
SELECT d.department
FROM department d
WHERE d.faculty = 'School of Computing';
```

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

   (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.

Comments:

One possible relational algebra expression.

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

Code: Alternative #1

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

An alternative is to use $\bowtie$ (i.e., `INNER JOIN`).

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

Code: Alternative #2: INNER JOIN

```sql
SELECT s.email
FROM student s
  INNER JOIN loan l ON (s.email = l.borrower OR s.email = l.owner)
                   AND l.borrowed < s.year;
```

We can also combine the $\bowtie$ with σ by moving some of the conditions around. This is fine as $\bowtie$ corresponds to `INNER JOIN`. Such technique may not be possible for `OUTER JOIN`.

Finally, we can use `UNION`.

$\pi_{[\texttt{s1.email}]}(\sigma_{[\texttt{s1.email=l1.borrower} \wedge \texttt{l1.borrowed<s1.year}]}(\rho(\texttt{student}, \texttt{s1}) \times \rho(\texttt{loan}, \texttt{l1})))$
$\quad \cup$

$\pi_{[\texttt{s2.email}]}(\sigma_{[\texttt{s2.email=l2.owner}\wedge\texttt{l2.borrowed<s2.year}]}(\rho(\texttt{student},\texttt{s2}) \times \rho(\texttt{loan},\texttt{l2})))$

Code: Alternative #3: UNION

```
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.borrower
  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.

Comments:
We use non-symmetric set difference. There are two convention for this operation. The first is $x-y$ and the second is $x\setminus y$. Both mean the same thing. Our convention is typically the first (i.e., $x-y$).

$\pi_{[\texttt{s1.email}]}(\sigma_{[\texttt{s1.email=l1.borrower}\wedge\texttt{l1.borrowed<s1.year}]}(\rho(\texttt{student},\texttt{s1}) \times \rho(\texttt{loan},\texttt{l1})))$
$-$
$\pi_{[\texttt{s2.email}]}(\sigma_{[\texttt{s2.email=l2.owner}\wedge\texttt{l2.borrowed<s2.year}]}(\rho(\texttt{student},\texttt{s2}) \times \rho(\texttt{loan},\texttt{l2})))$

Code: Query

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

2. **Universal Quantification.**

(a) Print the emails and the names of the different students who borrowed all the books authored by Adam Smith.

Comments:

We know one possible SQL query.

Code: Nested Query

```
SELECT s.email, s.name
FROM student s
WHERE NOT EXISTS (
  SELECT *
  FROM book b
  WHERE authors = 'Adam Smith'
    AND NOT EXISTS (
      SELECT *
      FROM loan l
      WHERE l.book = b.ISBN13
      AND l.borrower = s.email));
```

However, this SQL query contains `NOT EXISTS`, which is not directly translatable to relational algebra. What we can do is to break down the problem into the following subproblems.

1. Find the emails and names of the different students who have not borrowed at least one book authored by Adam Smith.
2. The solution is then the emails and names of the different students (including those who have not borrowed any books) *except* the students in Step 1.

Step 1 can be split even further.

a. Find all combinations of emails, names, and ISBN13 where emails and names are all students and ISBN13 are all books by Adam Smith.

b. Step 1's answer is then the result of Step a *except* the combinations for the students who borrowed books by Adam Smith.

c. We need to project the attributes from Step b to include only emails and names.

Putting everything together, we get the following relational algebra expressions. To ease reading, we add := symbol to split queries into smaller subqueries that we can refer to at a later time.

$Q_1 := \pi_{[\texttt{b1.ISBN13}]}(\sigma_{[\texttt{b1.authors='Adam Smith'}]}(\rho(\texttt{book}, \texttt{b1})))$

⇒ The ISBN13 of all books authored by Adam Smith.

$Q_2 := \pi_{[\texttt{s1.email},\texttt{s1.name}]}(\rho(\texttt{student}, \texttt{s1})) \times Q_1$

⇒ Relational algebra query for Step a. **Note:** The columns are `[email, name, ISBN13]`.

$$Q_3 := \pi_{[\texttt{s2.email},\texttt{s2.name},\texttt{b2.ISBN13}]}(\rho(\texttt{loan}, \texttt{l2}) \bowtie_{[\texttt{l2.book=b2.ISBN13} \wedge \texttt{b2.authors='Adam Smith'}]} \rho(\texttt{book}, \texttt{b2}) \bowtie_{[\texttt{l2.borrower=s2.email}]} \rho(\texttt{student}, \texttt{s2}))$$

⇒ Students who have borrowed books by Adam Smith.

$Q_4 \coloneqq \pi_{[\texttt{s2.email},\texttt{s2.name}]}(Q_2 - Q_3)$

$\Rightarrow$ Relational algebra query for Step b followed by projection. This is also the final answer for Step 1. **Note:** Q_2 and Q_3 are *union-compatible*. Now, the columns are `[email, name]`.

$Q_5 \coloneqq \pi_{[\texttt{s3.email},\texttt{s3.name}]}(\rho(\texttt{student}, \texttt{s3})) - Q_4$

$\Rightarrow$ **Note:** The projection on `s3` is to ensure *union-compatibility* with Q_4.

The answer is then Q_5.

The corresponding SQL query is shown below.

Code: Query from Relational Algebra

```sql
SELECT s3.email, s3.name
FROM student s3
EXCEPT
SELECT tmp.email, tmp.name
FROM (
  SELECT s1.email, s1.name, b1.ISBN13
  FROM student s1, book b1
  WHERE b1.authors = 'Adam Smith'
  EXCEPT
  SELECT s2.email, s2.name, b2.ISBN13
  FROM student s2, book b2, loan l2
  WHERE b2.authors = 'Adam Smith'
    AND s2.email = l2.borrower
    AND b2.ISBN13 = l2.book
) AS tmp;
```

References

[1] S. Bressan and B. Catania. *Introduction to Database Systems*. McGraw-Hill Education, 2006. ISBN: 9780071246507.

[2] Raghu Ramakrishnan and Johannes Gehrke. *Database Management Systems*. 2nd. USA: McGraw-Hill, Inc., 2000. ISBN: 0072440422.