IT5008: Tutorial 5 — Relational Algebra

Pratik Karmakar

School of Computing,
National University of Singapore

AY25/26 S1

Scenario

Students at the **National University of Ngendipura (NUN)** buy, lend, and borrow books.
NUNStA commissions *Apasaja Private Limited* to implement an online book exchange system that records:

- Students: name, faculty, department, **email**, join year.
- Books: title, authors, publisher, edition, **ISBN10**, **ISBN13**.
- Loans: `borrowed` date, `returned` date (`NULL` if active).

Auditing preserves records of graduated students and copies with loans.
This tutorial uses the schema/data from "SQL: Creating and Populating Tables."

Questions

1. Relational Algebra
   (a) Find the different departments in School of Computing.
   (b) Find emails of students who borrowed/lent a book *before* joining the University.
   (c) Find emails of students who borrowed but did not lend a book *on* joining day.
2. Universal Quantification
   (a) Find emails and names of students who borrowed *all* books authored by Adam Smith.

1(a). Departments in School of Computing

Relational Algebra:

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

SQL Equivalent:

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

1(b). Borrowed or lent before joining

Relational Algebra:

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

SQL Equivalent:

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

1(b). Alternative Forms

Using INNER JOIN:

```
1  SELECT s.email
2  FROM student s
3  INNER JOIN loan l
4  ON (s.email = l.borrower OR s.email = l.owner)
5  AND l.borrowed < s.year;
```

Using UNION:

```
1  SELECT s1.email
2  FROM loan l1, student s1
3  WHERE s1.email = l1.borrower
4    AND l1.borrowed < s1.year
5  UNION
6  SELECT s2.email
7  FROM loan l2, student s2
8  WHERE s2.email = l2.owner
9    AND l2.borrowed < s2.year;
```

1(c). Borrowed but did not lend on joining day

Relational Algebra:

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

SQL Equivalent:

```
1  SELECT s1.email
2  FROM loan l1, student s1
3  WHERE s1.email = l1.borrower
4    AND l1.borrowed = s1.year
5  EXCEPT
6  SELECT s2.email
7  FROM loan l2, student s2
8  WHERE s2.email = l2.owner
9    AND l2.borrowed = s2.year;
```

2(a). Borrowed all books by Adam Smith

SQL Nested Query:

```sql
SELECT s.email, s.name
FROM student s
WHERE NOT EXISTS (
  SELECT 1
  FROM book b
  WHERE b.authors = 'Adam Smith'
    AND NOT EXISTS (
      SELECT 1
      FROM loan l
      WHERE l.book = b.ISBN13
        AND l.borrower = s.email));
```

Idea: For each student, check that there is no Adam Smith book that they have not borrowed.

2(a). Relational Algebra Strategy

Break the problem into steps (universal quantification via set difference):

1. **Find all Adam Smith books.**

$$Q1 = \pi_{\text{ISBN13}}(\sigma_{\text{authors} = \text{'Adam Smith'}}(\text{book}))$$

2. **Form all possible student–book pairs (expected borrowings).**

$$Q2 = \pi_{\text{email, name, ISBN13}}(\text{student} \times Q1)$$

3. **Find actual borrowings of Adam Smith books.**

$$Q3 = \pi_{\text{email, name, ISBN13}}(\text{student} \bowtie \text{loan} \bowtie \sigma_{\text{authors}=\text{'Adam Smith'}}(\text{book}))$$

4. **Identify missing borrowings (pairs in Q2 but not in Q3).**

$$Q4 = Q2 - Q3$$

5. **Remove students in Q4 from the set of all students.**

$$Q5 = \pi_{\text{email, name}}(\text{student}) - \pi_{\text{email, name}}(Q4)$$

Answer: $Q5$ = students who borrowed *all* Adam Smith books.

2(a). SQL from Relational Algebra

```sql
SELECT s.email, s.name
FROM student s
EXCEPT
SELECT t.email, t.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
       JOIN loan l2 ON s2.email = l2.borrower
       JOIN book b2 ON b2.ISBN13 = l2.book
  WHERE b2.authors = 'Adam Smith'
) AS t;
```

Guidelines & Remarks

- Use relational algebra to reason about SQL queries.
- Remember: EXCEPT in SQL corresponds to set difference () in algebra.
- Joins in SQL correspond to natural/-joins in algebra.
- Break universal quantification into difference-based subqueries.
- Keep queries readable: aliases, indentation, uppercase SQL keywords.

Questions?
Drop a mail at: pratik.karmakar@u.nus.edu