CS4221/5421: Tutorial 2 — Data Warehousing

Pratik Karmakar

School of Computing,
National University of Singapore

AY25/26 S2

The Problem Statement

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 his/her email. The database also records the date at which the student joined the university. If a student has graduated, the database record the date of graduation. A department in NUN must belong to exactly one faculty.
The database records the title, authors, publisher, language, year as well as the ISBN-10 and ISBN-13 for each book. A book can have several authors but it must have at least one author. The database also records author that currently has no book. It should also record the format of the book (i.e., if the book is hardcover or softcover). 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 student (e.g., 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).
A student may own multiple copies of the same book. We differentiate the copy by its copy number. For instance, John may own two copies of the book Database Systems with ISBN-13 number of 9780131873254. The first copy has a copy number of 1 while the second copy has a copy number of 2. The copy number should be a consecutive number starting from 1.
The database also 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. Obviously, a student can only borrow or lend book after he/she is enrolled.
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.

Questions

1. **OLTP**
    1. For each owner faculty and borrower faculty pair, find the average duration, and the longest duration such that the owner is from the different faculty than the borrower.
       Output the owner faculty, borrower faculty, average duration, and longest duration. Round the average duration to 2 decimal places using `ROUND(val, dp)`. Sort the output in descending order of average duration.
    2. For each book, find the student(s) that borrow the book for the longest duration. Output the book title and the name of the student. Sort the result in ascending order of book ISBN13 and descending order of student email.
       Multiple students may borrow a book for the longest time if they have the same longest time. In such cases, output all such students with the longest duration for the book.
2. **Transform**
   Transform the OLTP database into an OLAP database (star schema) supporting Q1.
3. **OLAP**
   Redo Q1 on the star schema.

Loan duration definition

We define **loan duration** as:

- if the item has been returned: `rd_date` − `bd_date`
- otherwise: `CURRENT_DATE` − `bd_date`

In SQL, this is typically implemented using `COALESCE(...)` to convert `NULL` return dates into a meaningful duration.

Q1(a) OLTP solution

For each owner faculty and borrower faculty pair (different faculties), compute average and longest loan duration, and the longest duration such that the owner is from the different faculty than the borrower:

```
SELECT d1.d_faculty AS owner_faculty,
       d2.d_faculty AS borrower_faculty,
       ROUND(AVG(COALESCE(l.rd_date - l.bd_date,
                          CURRENT_DATE - l.bd_date)), 2) AS average,
       MAX(COALESCE(l.rd_date - l.bd_date,
                    CURRENT_DATE - l.bd_date)) AS longest
FROM loan l, students s1, students s2, departments d1, departments d2
WHERE l.s_owner = s1.s_email
  AND l.s_borrower = s2.s_email
  AND s1.d_name = d1.d_name
  AND s2.d_name = d2.d_name
  AND d1.d_faculty <> d2.d_faculty
GROUP BY d1.d_faculty, d2.d_faculty
ORDER BY average DESC;
```

Q1(a) OLTP notes

- **Repeated computation:** the duration expression appears in both `AVG` and `MAX`.
- **Why `COALESCE`?** `rd_date` can be `NULL` if the book is not yet returned.
- **Complexity:** the query joins multiple tables and then groups by faculty pairs; runtime is dominated by join + grouping (and can benefit from indexes on join keys).

Q1(b) OLTP solution

For each book, output the student(s) who borrowed it for the *longest duration*:

```sql
SELECT b.b_title AS title, s.s_name AS name
FROM books b, students s, loan l
WHERE s.s_email = l.s_borrower
  AND b.b_isbn13 = l.b_isbn13
GROUP BY b.b_isbn13, s.s_email, b.b_title, s.s_name
HAVING MAX(COALESCE(l.rd_date - l.bd_date,
                    CURRENT_DATE - l.bd_date)) >= ALL (
  SELECT MAX(COALESCE(l1.rd_date - l1.bd_date,
                      CURRENT_DATE - l1.bd_date))
  FROM books b1, students s1, loan l1
  WHERE s1.s_email = l1.s_borrower
    AND b1.b_isbn13 = l1.b_isbn13
    AND b1.b_isbn13 = b.b_isbn13
  GROUP BY b1.b_isbn13, s1.s_email
)
ORDER BY b.b_isbn13 ASC, s.s_email DESC;
```

Q1(b) OLTP notes

- This pattern uses `>= ALL (subquery)` to keep the borrower(s) achieving the per-book maximum duration.
- Multiple students can be returned for a book if they tie on the longest duration.
- In practice, this can be visibly slow on larger loan tables (e.g., seconds for a few thousand rows), depending on machine and indexes.

Star schema design (for Q1)

- **Measure:** loan duration.
- **Dimensions:**
  - students as **owners**
  - students as **borrowers**
  - **books**
- We create two (pretty much identical) dimension tables for owners and borrowers to enable natural joins in the OLAP queries.

Example: INSERT with query

Copying a department table into the warehouse schema:

```sql
CREATE TABLE wh_department (
  d_dept    VARCHAR(32) PRIMARY KEY NOT NULL,
  d_faculty VARCHAR(62) NOT NULL
);

INSERT INTO wh_department (
  SELECT d_name, d_faculty
  FROM departments
);
```

Q3(a) OLAP solution

Repeat Q1(a) on the star schema:

```sql
SELECT o_faculty AS borrower_faculty,
       b_faculty AS owner_faculty,
       AVG(l_duration) AS average,
       MAX(l_duration) AS longest
FROM wh_loans
NATURAL JOIN wh_owners
NATURAL JOIN wh_borrowers
NATURAL JOIN wh_books
WHERE o_faculty <> b_faculty
GROUP BY o_faculty, b_faculty
ORDER BY average DESC;
```

Q3(a) OLAP notes

- The query is **simpler to write** due to the star schema.
- Speed may be similar on small data, but the schema is designed to scale better for analytic workloads.

Q3(b) OLAP solution

Repeat Q1(b) on the star schema:

```
SELECT k_title AS title, o_name AS name
FROM wh_loans
NATURAL JOIN wh_owners
NATURAL JOIN wh_borrowers
NATURAL JOIN wh_books k1
GROUP BY k_isbn13, o_email, k_title, o_name
HAVING MAX(l_duration) >= ALL (
  SELECT MAX(l_duration)
  FROM wh_loans
  NATURAL JOIN wh_owners
  NATURAL JOIN wh_borrowers
  NATURAL JOIN wh_books k2
  WHERE k1.k_isbn13 = k2.k_isbn13
  GROUP BY k_isbn13, o_email, k_title, o_name
)
ORDER BY k_isbn13 ASC, o_email DESC;
```

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