Session 5

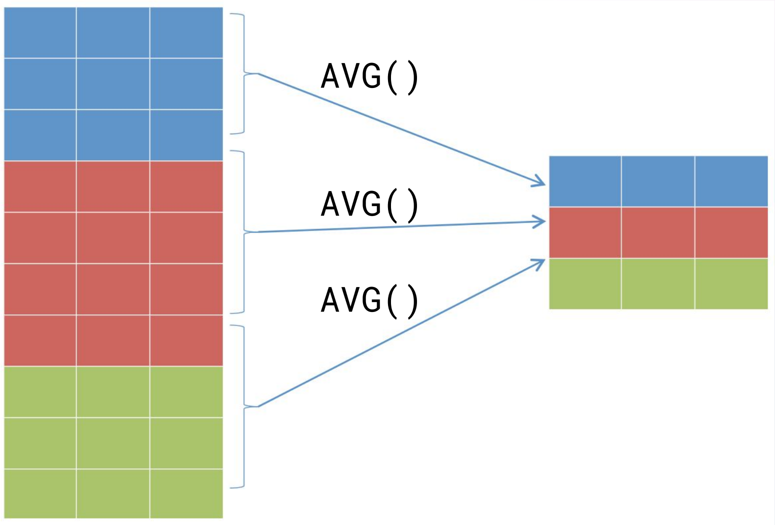
Relational Queries – Part 2

Normalization

sourcemind

Aggregate functions

- Perform calculation on a group of rows and return a single row as the result
- `AVG(), COUNT(), SIM(), MIN(), MAX()`
- Used with `GROUP BY` keyword



GROUP BY

- Count number of books in each topic

```
SELECT topic_id, count(*)
FROM book
GROUP BY topic_id;
```

- To include topic name, we need to perform a join

```
SELECT topic_name, count(*)

FROM book NATURAL JOIN topic_copy

GROUP BY topic_name;
```

HAVING

- To filter group and aggregation result, HAVING keyword is used
- HAVING is similar to WHERE clause
- WHERE is applied before grouping, HAVING is applied after grouping

```sql
SELECT topic_name, count(*)
FROM book NATURAL JOIN topic_copy
GROUP BY topic_name
HAVING count(*) > 1;
```

UNION, UNION ALL

- (Query result 1) UNION (Query result 2) ⇒ combines the two results, removes duplicates
- (Query result 1) UNION ALL (Query result 2) ⇒ keeps duplicates
- Query 1 and query 2 results must have the same schema

```sql
SELECT title
FROM book
WHERE topic_id = 1
UNION
SELECT title
FROM book
WHERE topic_id = 2;
```

EXCEPT, INTERSECT

- Exclude the results of Query 1 from Query 2
- Find the intersection of Query 1 and Query 2 results

```sql
SELECT title
FROM book
EXCEPT
SELECT title
FROM book
WHERE topic_id = 2;
```

Common Table Expressions (CTE)

- A calculated expression to use in another query
- As soon as the query is executed the CTE is gone

```
WITH book_category AS (
    SELECT book.title, t.topic_name AS topic
    FROM book
             INNER JOIN topic t on t.id = book.topic_id
)
SELECT *
FROM book_category
WHERE topic = 'Programming';
```

Recursive queries

- Create a hierarchical task structure

```
Task 1
    Task 1-1
        Task 1-1-1
        Task 1-1-2
    Task 1-2
        Task 1-2-1
Task 2
```

```
CREATE TABLE task
(
    task_id   INT PRIMARY KEY NOT NULL,
    title     VARCHAR(100)    NOT NULL,
    parent_id INT DEFAULT NULL
);
```

```
INSERT INTO task
VALUES (1, 'Task 1', null),
       (2, 'Task 1-1', 1),
       (3, 'Task 1-2', 1),
       (4, 'Task 1-1-1', 2),
       (5, 'Task 1-1-2', 2),
       (6, 'Task 1-2-1', 3),
       (7, 'Task 2', null);
```

Select a parent task is simple...

```sql
SELECT *

FROM task;


-- Find the parent of Task 1-1-2

SELECT *

FROM task
WHERE title = 'Task 1-1-2';
```

Select children of task with task_id = 1

- Recursive CTE
- Base clause (simple case)
- Recursive clause
- Union

```
WITH RECURSIVE subtasks AS (
    -- Base clause
    SELECT *
    FROM task
    WHERE task_id = 1

     UNION ALL

     -- Recursive clause
    SELECT t.*
    FROM task t
             INNER JOIN subtasks s ON s.task_id = t.parent_id
)
SELECT *
FROM subtasks;
```

Normalization

- Eliminate data redundancy
- Having logical data dependencies
- Avoid potential data anomalies

A non-normalized table: books_borrows

username	name	borrow	major	topic
alice	Alice	1, 2, 4	AI	DB, Java, Math
bob	Bob	1, 3	Physics	DB, Calculus
jon	Jon	2, 4	SW Eng	Java, Math

1NF: 1st normal form

- No multivalued columns allowed
- rented_books: 1, 2, 4 is not single-valued

- Normalization:

  Repeat the multivalued values of a column in new rows such that each column has only one value

A non-normalized table

username	name	borrow	major	topic
alice	Alice	1	AI	DB
alice	Alice	2	AI	Java
alice	Alice	4	AI	Math
bob	Bob	1	Physics	DB
bob	Bob	3	Physics	Calculus
jon	Jon	2	SW Eng	Java
jon	Jon	4	SW Eng	Math

Identifying the PK

- What is the primary key in **books_borrows** table?

username	**name**	**borrow**	**major**	**topic**

- We have a **composite key**: (username, borrows)

Functional Dependency

- There is functional dependency (DF) between X and Y, if for every row with a value in X there is the same value in Y (FD: X → Y)
- If two rows have same value of X then they should also have same value for Y

- username → name
- username → major
- borrow → topic
- username, borrows → name, major, department
- ....

username	name	borrow	major	topic
alice	Alice	1	AI	DB
alice	Alice	2	AI	Java
alice	Alice	4	AI	Math
bob	Bob	1	Physics	DB
bob	Bob	3	Physics	Calculus
jon	Jon	2	SW Eng	Java
jon	Jon	5	SW Eng	Math

2NF: 2nd normal form

- Must be in 1NF
- There must not be any partial dependency of any column on primary key.
- If there is a composite PK (username, borrows), then each column that is not in the PK must depend on entire composite key.
- Violations:
  - username → name (name depends on part of PK)
  - borrow → topic (topic depends on part of PK)

- Normalization:
  Split the table
  T1 (username, name, borrow)
  T2 (borrow, topic)

2NF split

username	name	borrow	major	topic
alice	Alice	1	AI	DB
alice	Alice	2	AI	Java
alice	Alice	4	AI	Math
bob	Bob	1	Physics	DB
bob	Bob	3	Physics	Calculus
jon	Jon	2	SW Eng	Java
jon	Jon	5	SW Eng	Math

borrow	topic
1	DB
2	Java
3	Math
4	Calculus

username	name	borrow	major
alice	Alice	1	AI
alice	Alice	2	AI
alice	Alice	4	AI
bob	Bob	1	Physics
bob	Bob	3	Physics
…	...	...	...

Repeat the split until 2NF holds

3NF: 3rd normal form

- Must be in 2NF
- No transitive dependency: No dependency on non-key attributes

username	**name**	**major**	**department**

- The OK is username
- There is a FD: major → department
- Normalization:
  Split the table so that there are FDs only on PKs

3NF split

username	name	major	department

username	name	major

major	department

Overview

username	name	major

major	department

borrow	username

borrow	topic

Homework 4

- Create a hierarchy of topics

```
IT Books
     |----- Programming
          |------ Java
          |------ C++
     |----- Databases
          |------ Relational Databases
               |------ Oracle
               |------ Postgres
          |------ NoSQL Databases
               |------ MongoDB
```

- Add some data (books and topics)
- Select all books from the `Programming` category
- Select all books from the `Relational Databases` category
- Get the number of books in the `IT Books` category
- Write all SQL commands in a single file and attach that in the homework submission