Advanced Database Management

Project Report

Team: Influence
Project repository: https://github.com/yerlan2/Advanced-DBMS-Project
Technology stack of the project:

- Programming language: Python (Flask)
- Database: SQL (SQLite)

1. Idea

TL/DR: Our project pretty resembles medium.com.

Influence is a community publishing platform, blurring the lines between blogging and social networking. This is a web application. It is home to a wide array of creative individuals looking to share common interests, meet new friends, and express themselves.

Influence encourages communal interaction and personal expression by offering a user-friendly interface.

2. Dataset

We generated fake data by the means of python libraries faker and numpy.
All generated data was first stored in structured pandas dataframes and then were converted into .csv files.
Data is here. You can access the code through this link.

3. E/R Diagram and database modelling

Users of Influence may query information about articles by article title, content and category name. It allows users to log in to the system and make various queries using the components of the UI. Once the user selects parameters in the UI, his choices are translated into a corresponding SQL query string. The query string is passed on to Python which issues the actual query to the Influence database.

Influence database has the following entities:

ACCOUNT: This entity models Influence’s users. It stores user details such as the user id, image id, name, email and password. The user's id is used as the primary key. The image id is used as the foreign key and specifies the avatar of the user.

POST: This entity stores details of articles such as post id, account id, category id, title, content, date, like amount, view amount. The post id is the primary key used to identify the article. By default, each post has zero likes and number of views.

COMMENT: This entity models Influence’s users comments given to a specific post and stores details such as comment id, post id, account id, content, date. A post may have no comments or have many comments, but a comment without a post does not exist.

LIKE: This entity stores details such as post id and account id. The entity doesn’t have a primary key and is used to calculate the number of likes of the post and the owner of the like.

FOLLOW: This entity stores details such as follow id and account id. The entity serves to find out how many subscribers the author has.

CATEGORY: This entity stores details such as category id and name. The category id is used as the primary key. Each post must have its own category. This entity helps users quickly find the necessary records and helps them easily navigate the site.

IMAGE: This entity stores details such as image id and path. The image id is the primary key. Each image can be used in multiple posts.

POST_IMAGE: This entity models Influence’s posts images. It stores post id and image id. Each post has an image, it can be one or more.

4. SQL queries

Based on ideas and requirements, we started writing sql queries. Selecting posts from the user oneself and other users to which user is subscribed, posts from other users, posts with a specific category, certain posts. search posts, accounts, followers, users the user is following. Inserting and deleting likes, follows, comments, posts and images. Updating posts and images.

5. UI implementation

We used ready design solutions.
Most of our website pages are taken from Meranda — Website Template by Colorlib.
Data visualization page is built using chart.js.

6. Calculating true cardinalities and plan evaluation, performance evaluation

We created SQL queries to get useful information in the database. To get useful information, it is impossible to work with only one table, at least with our database. Since there are 8 tables in our database, interacting with each other with the wrong layout will take a lot of time. We needed query optimization.

First, we evaluated the existing requests by calculating the true cardinality and time to complete the request. To find out where queries are spending more time and why, we moved our database to Postgresql and used the "explain analysis" command.

After that, we used the knowledge gained during the course to query optimization. We planned the process by using a relational algebra expression tree and calculating the true cardinality of queries by combining tables with different methods. And summing up the result with a check of the "explain analysis". As a result, optimized queries run faster than the old ones.

7. Challenges and decisions

Before we used our current technical stack we planned to use Django framework for easy development and learning purposes. But it later appeared that it uses O/RM (Object–relational mapping) which is not suitable for this course. So, we switched to Flask which is nice as well.

Also notice that, since the project has a lot of data and more than 7,000 posts, it becomes very difficult to browse and it becomes unproductive to process all the data every time, so we added ***pagination***. With it we divide a large amount of data into several small parts and show them in parts.