# Duration

24-32 hours

# Objective

To learn the basics of relational databases and SQL-language with the help of database management PostgreSQL

# Materials for learning

1. The official documentation PostgreSQL <https://www.postgresql.org/docs/10/static/index.html>
2. Tutorial PostgreSQL <http://www.postgresqltutorial.com/>
3. Indexes in PostgreSQL  <https://habr.com/company/postgrespro/blog/326096/>
4. Transaction isolation levels with examples on PostgreSQL  <https://habr.com/post/317884/>
5. The book Martin Gruber "Understanding SQL"
6. Tutorial SQL <https://www.w3schools.com/sql/default.asp>
7. Fundamentals of Database Design  <https://www3.ntu.edu.sg/home/ehchua/programming/sql/Relational_Database_Design.html>
8. The cheat sheet about PostgreSQL  <http://www.postgresqltutorial.com/wp-content/uploads/2018/03/PostgreSQL-Cheat-Sheet.pdf>
9. PostgreSQL locking tips <https://dzone.com/articles/when-postgres-blocks-7-tips-for-dealing-with-locks>
10. An Introduction to B-Trees: <https://www.youtube.com/watch?v=C_q5ccN84C8>
11. Window function documentation (PostgreSQL 13): <https://postgrespro.ru/docs/postgresql/13/functions-window>
12. Examples of the use of window functions: <https://habr.com/ru/post/268983/>
13. Reading EXPLAIN at maximum speed <https://habr.com/ru/company/citymobil/blog/545004/>

# Tasks

### **Task 1**

To complete the tutorial PostgreSQL (see materials to use point 2)

### **Task 2**

To design the database schema for the following application:

* You should store the following data for each user: name, surname, date of birth, email, password, addresses of residence.
* Each user can publish posts and set up their titles, content, tags and status. Each post can have the following statuses: published, draft or in archive.
* Users can edit the posts that were created by other users.
* Each user can like and dislike the posts from other users.
* Each user can comment on the post from other users. The comment contains only the text.
* Each user can like and dislike the comments from other users.
* Statistics of visits by the day are stored for each post.
* Each post has a rating: number of likes minus number of dislikes.
* Each user has a rating: 50% is the average rating of the posts that user created, 30%  is the average rating of the posts that user edited, 20% is the average rating of his comments.

To write the script for completion of the data sheets with test data. There should be at least 100000 posts for different users. You can use the function [generate\_series](https://www.postgresql.org/docs/9.6/functions-srf.html) to generate data.

|  |
| --- |
| For tables with next structure  CREATE TABLE users  (  id SERIAL PRIMARY KEY,  name VARCHAR(255) NOT NULL,  email VARCHAR(255) UNIQUE NOT NULL  );  CREATE TABLE posts  (  id SERIAL PRIMARY KEY,  author\_id INTEGER NOT NULL,  title VARCHAR(255) NOT NULL,  text TEXT NOT NULL,  status INTEGER NOT NULL,  created\_at TIMESTAMP NOT NULL DEFAULT NOW(),  FOREIGN KEY (author\_id) REFERENCES users (id) ON UPDATE CASCADE ON DELETE RESTRICT  );  You can use this procedure (It creates 10000 users, with 50 постов at each, every 10nth post "awaits moderation" (status=2), the rest are "published" (status = 3)):  DO $$  DECLARE v\_users\_number INT;  DECLARE v\_posts\_for\_each\_user INT;  BEGIN  v\_users\_number := 10000;  v\_posts\_for\_each\_user := 50;  INSERT INTO users SELECT  num,  concat('name', num),  concat('email', num, '@example.com')  FROM generate\_series(1, v\_users\_number) as num;  INSERT INTO posts SELECT  num,  (num - 1) % v\_users\_number + 1,  LEFT(MD5(num::varchar), 10),  MD5(num::varchar),  3,  NOW()  FROM generate\_series(1, v\_posts\_for\_each\_user \* v\_users\_number) as num;  UPDATE posts SET status = 2 WHERE id % 10 = 0;  END $$; |

### Task 3

(Obligatory)

To write the following queries to DB:

* To count the number of the posts for the user with the specified ID;
* To select N published posts that are sorted in descending order of creation date;
* To select N posts in status "waiting to be published", that are sorted in ascending order of creation date;
* To find N recently updated posts with the specified tag for K page (there are L posts in each page);
* To find N posts with the highest rating for day/month/year.

To estimate time for executing a query (on enough test data) and analyze query execution plans.

To reduce the time for executing queries using appropriate indexes.

To compare time for executing queries and query execution plans after creating indexes.

To estimate the size of the indexes used. If possible to reduce the size of the created indexes.

### Task 4

To write the following queries to DB:

* To find N most visited posts for day/month/year.
* To find N most visited posts for the specified user that were edited but not created by this user for all time.
* To find N users for whom the total rating for all posts created by them is the highest among all users.
* To find N users for whom the total rating for all posts created by them is the highest among all users under K years old.
* To find N users with the highest rating.
* To find N tags for which the total number of visits to their related posts is the largest in a week.

### Task 5

(Optional)

To optimize queries from Task 4. To complete a detailed report with the optimization results: reasons for long query execution, solutions, reasons why a specific solution was chosen, comparison of the time for executing queries before and after optimization.

### Task 6

To write the following queries to DB using transactions:

* To make a copy by id with related authors and tags but without statistics, comments and rating. The copied post should be in status “draft”.
* To remove all the users who have rating lower than N together with all the posts and comments. The order of  removing the entities: user’s comments to posts, user’s comments, user’s posts, user.

To add the following columns to the table using alter table:

* Users status. The status can have values like “active” and “blocked”. The new column should be filled with the “active” value.
* The dates of creation and updating posts. The new columns should be filled with the values of the current date.

### Task 7

To add several articles (at least 20) that go in order of dates with different skips, for example:

2021-07-01, 2021-07-05, 2021-07-06, 2021-07-07, 2021-07-10 and so on.

For several users (at least 3).

To add tags for each article (a random number).

1. To display the rating by number of tags under the users articles in ascending order of  tags number.
2. To display the total accumulated number of tags to the current date for each user.
3. To add the columns to task 2 with the following values: the amount of tags in the first day, the previous day, the next day and the last day of the period.
4. Optionally: To supplement task 3 with other window functions from documentation.

### **Obligatory requirements**

* When creating the tables you should limit the possible values of each field as much as possible (unique, foreign key etc.)
* To use indexes.
* Adding of the new columns to the table should not block it and its’ recordings.
* Adding of the new index to the table should not block it and its’ recordings.