Exercise Sheet 8: SQL I (until Thursday, 14.12.2017) (45 Points)

Please note: you need **50%** of all exercise points to receive the *Studienleistung* for this lecture. In order to pass the RDB I Module, you need both the Studienleistung **and** you need to pass the exam. Exercises have to be turned in until **Thursday before the lecture** either in the lecture hall or into our mailbox at the IFIS floor (Mühlenpfordtstraße 23, 2nd floor). Please do not forget your **Matrikelnummer** and your **tutorial group number** on your solutions. **If you forget** to write your Matrikelnummer and/or your tutorial group number, you get **automatically 0 points**. Your solutions may be in German or English. Unless otherwise specified: **Always use your own words!**

PLEASE: THIS HOMEWORK MUST BE SEND TO YOUR HIWI BY EMAIL

General Information

This week we will start using SQL!. There are different options in terms of software that you can use. We suggest you to use one of the most exciting open source database for our class: PostgreSQL.

Tasks

1.-Please download and install PostgreSQL from: http://www.postgresql.org/

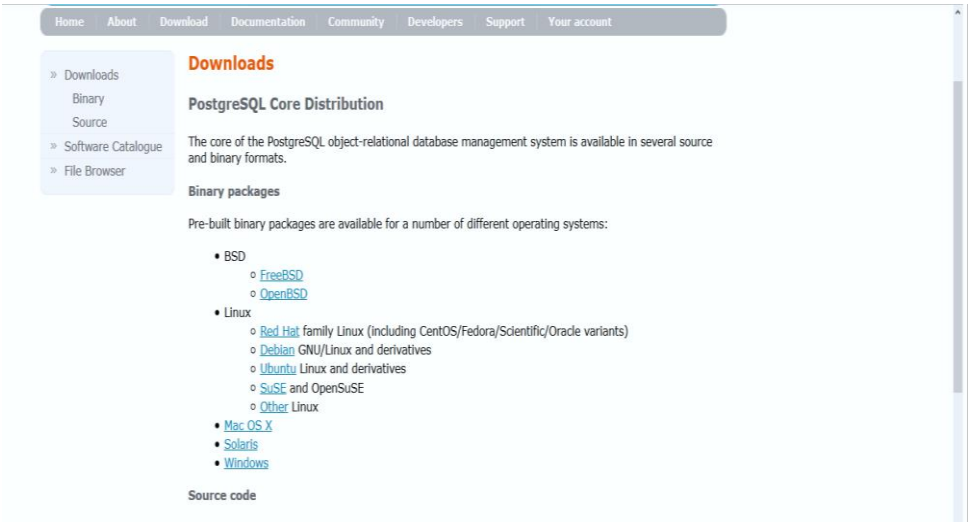

Figure 1 PostgreSQL download page

The installer should provide you with a straightforward installation process. After an effective installation, you should be able to use **pgAdmin III**; a graphical tool for managing and developing your databases.

2.-Please familiarize yourself with **pgAdmin III**. In particular, you need to know how to create a new database, how to restore a database from a backup, and how to open the **SQL editor** to run your queries. Figure 2 show a screen shot of a running instance of **pgAdmin III.**

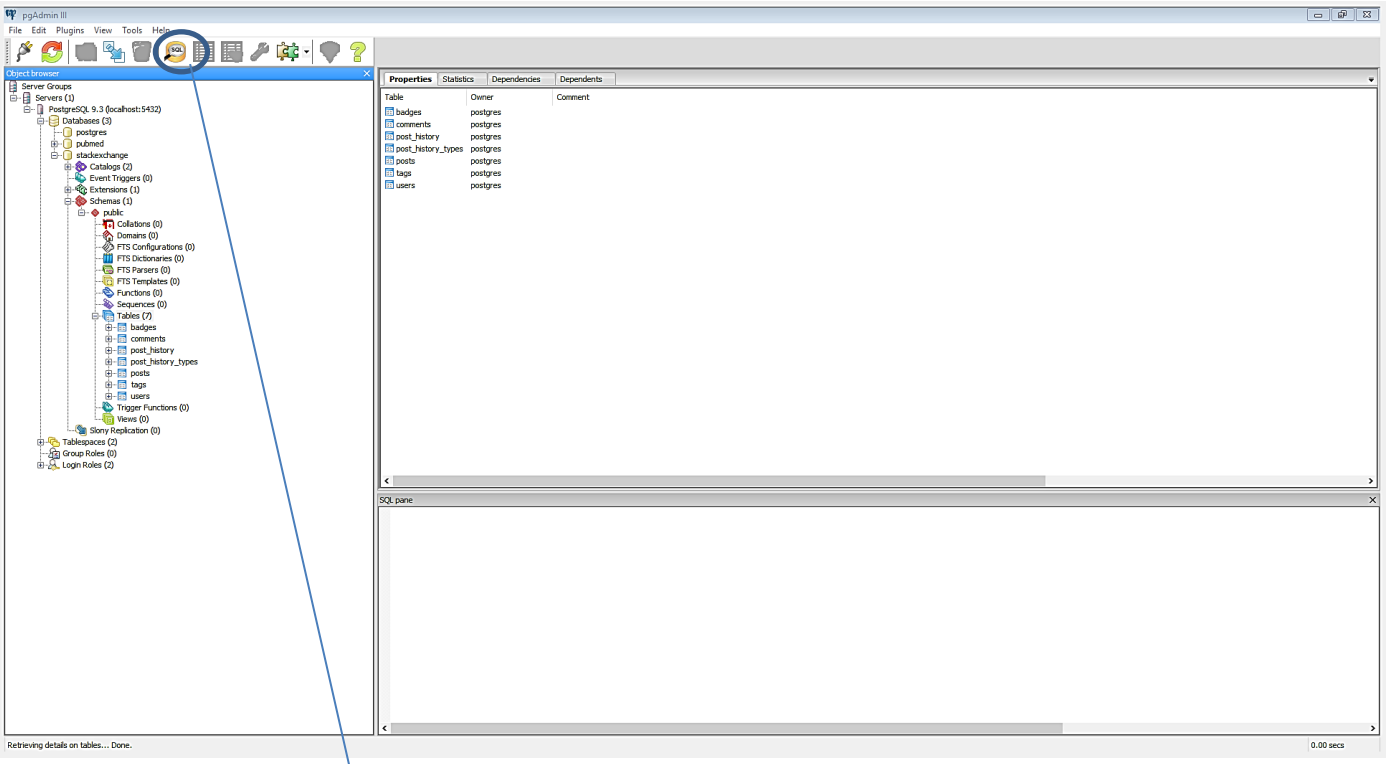

Figure 2. pgAdmin III

Once you click on the **SQL Editor Button**, you should get your SQL editor, where your will be running your queries. Figure 3 shows a screen shot.

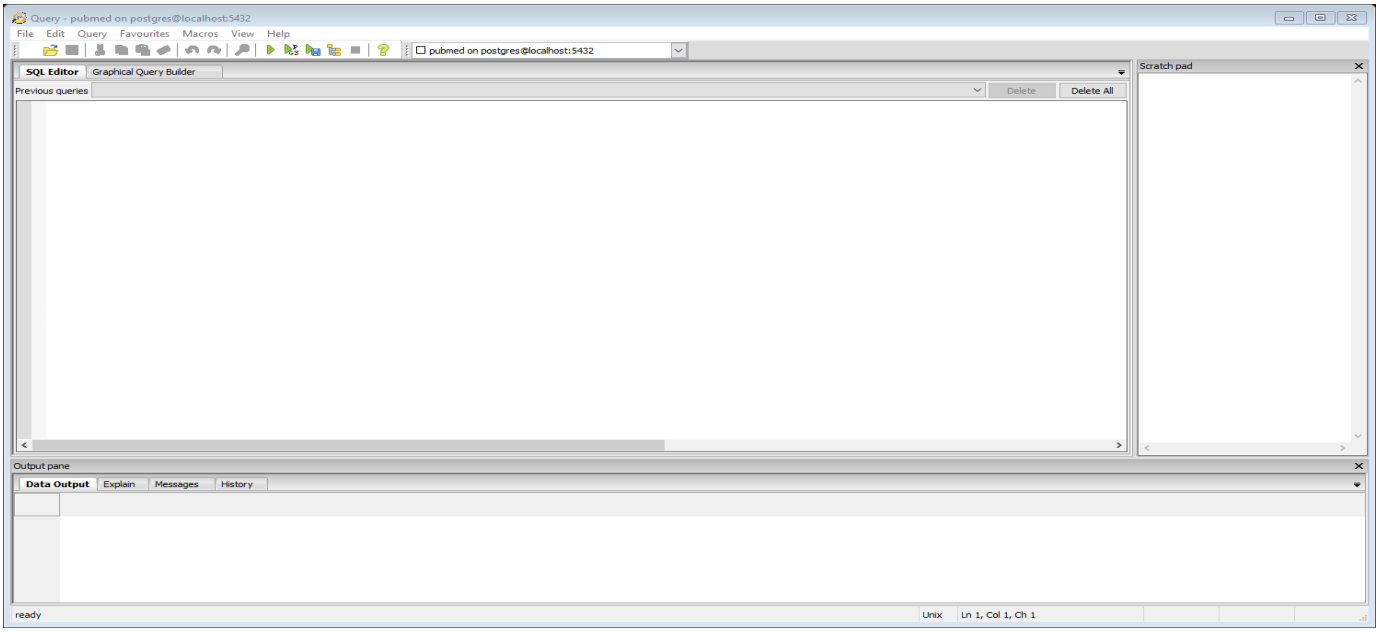

Figure 3 SQL Editor

3.-Use the **pgAdmin III** interface to create a new database and load data. Please proceed as follows:

- Download the data from the following link: http://www.ifis.cs.tu-bs.de/webfm_send/1910.

- Create a new database. The database name must be: "stackexchange".
- Click on the database you just created and select the *Restore* option from the context menu as shown in the following screen:

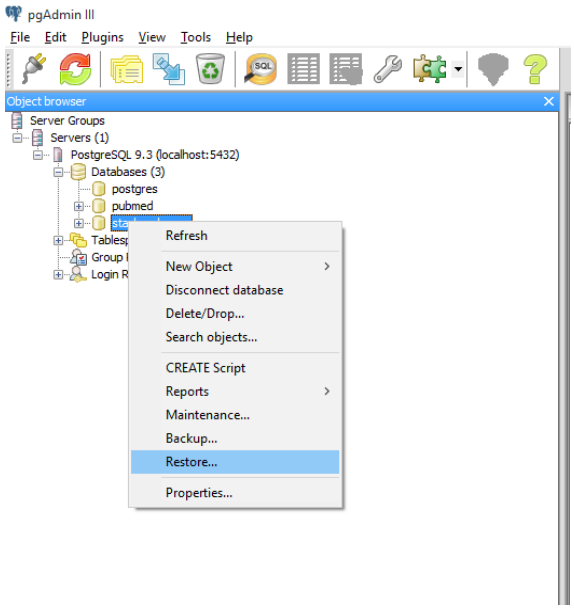

Figure 4. Restore option from the context menu.

- After you have selected the "Restore" option from the context menu, a screen similar to the one shown in Figure 5 will allow you to complete the upload of the data to the database. First, you will need to specify the path in your computer where the downloaded data has been stored. Next, click on the *Restore* button to finish the task and you will be ready to start writing your queries!

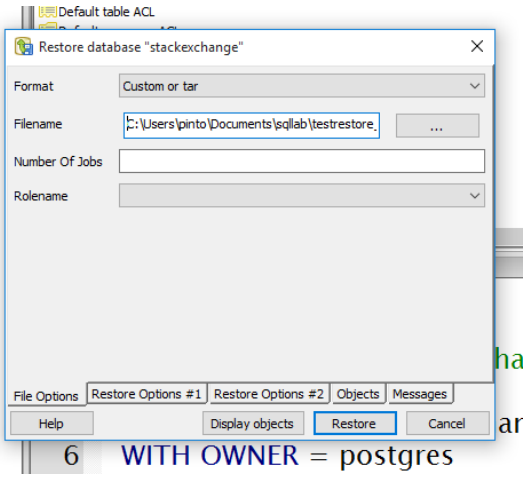

Figure 5. Restore

Your first task is to answer queries using SQL on a database that corresponds to a simplified version of Stack Exchange (http://stackexchange.com). In case you have never used Stackexchange, please take the tour (http://stackexchange.com/tour) to get an idea of the different elements you need to consider. In particular, consider the following:

Stackexchange is a network of Q&A of several communities that are created and run by experts and enthusiasts who are passionate about a specific topic. The database that you are going to use corresponds to one community and the schemas are:

- **Posts**(post_id:integer, post_type:integer, title:string, accepted_answer :integer, parent_id:integer, owner_user_id:integer, score:integer, day:integer, month:integer, year:integer)
- **Users**(id:integer, reputation:integer, display_name:string, day:integer, month:integer, year:integer, location:string, up_votes:integer, down_votes:integer, age:integer)
- **Comments** (id:integer, user_id:integer, post_id:integer, score:integer, text:string, day:integer, month:integer, year:integer)
- **Badges**(id:integer, user_id:integer, name:string, day:integer, month:integer, year:integer)
- **PostHistory**(id:integer, post_history_type_id:integer, post_id:integer, day:integer, month:integer, year:integer, user_id:integer)
- **PostHistoryTypes**(id:integer, name:string)
- **PostsTags**(post_id:integer, tag_id:integer)
- **Tags**(id:integer, tag_name:string)

Explore the database, use SQL commands to understand the structure of the data. Can you guess to which specific community of stack exchange this database corresponds to? Here are some clarifications of the schema:

Posts: every Post has a unique Id. Posts are of different types. For this exercise the following two types are important: questions=1, answers=2; just keep in mind that there are other types in the database.

- If the post is an answer, then the attribute accepted_answer is 0. If the post is a question, but has no accepted answer yet, then accepted_answer is -1.
- Every post has a creation date, and a score count that reflects the popularity of the post (the higher the better).
- Every post has a title and a set of tags that further describe the post (tags per post are in the table PostsTags).
- For the attribute parent_id: it is always 0 for posts that are of type question; and for posts of type answer it contains the post ID of the question that this posts attempts to answer. In other words, this attribute allows one to know the possible answers for a given post of type question.

Users: users have unique Id. Some other characteristics relevant to the users table are: reputation (the higher the better), date of creation of the account, name, location, age. The attributes up_votes and down_votes have the number of times a user has voted positive or negative, respectively, to any post.

Comments: users can make comments about posts. Every comment has a unique id, a score, the post id that corresponds to the comment, the text of the comment, creation date, and the user that made the comment.

Badges: these are special achievements that a user can earn for participating on the site. Each Badge has a unique id, a name, the id of the user that has the badge, and the date of creation.

Tags: tags help to describe a post. Tags are subject areas (keywords or phrases).

PostHistoryTypes: This table has a catalog that describes different status of each post as the post is edited through different community members. The following types are in our toy database:

The following are the post history types (PostHistoryTypes):

Id	Name (description)
1	Initial Title - The first title a question is asked with.
2	Initial Body - The first raw body text a post is submitted with.
3	Initial Tags - The first tags a question is asked with.
4	Edit Title - A question's title has been changed.
5	Edit Body - A post's body has been changed, the raw text is stored here as markdown.
6	Edit Tags - A question's tags have been changed.
7	Rollback Title - A question's title has reverted to a previous version.
8	Rollback Body - A post's body has reverted to a previous version - the raw text is stored here.
9	Rollback Tags - A question's tags have reverted to a previous version.
10	Post Closed - A post was voted to be closed.

PostHistory: this table has all the different post history types that a post may have. It has a unique id, a post history type id, the creation date, and the user id responsible for the entry. Using the stack exchange database answer the following queries using SQL.

Excercise 8.1 (27 Points)

1. **(3 Points)** Give the percentage of acceptance rate of the user with id =10. In other words, on average how often are answers from user id=10 accepted?
2. **(3 Points)** How many users have not contributed with any type of post?
3. **(3 Points)** For the user with id=101, list the name of the most frequent tag he has used.
4. **(3 Points)** Show the title, the post Id and the score of the question with the highest score.
5. **(3 Points)** List the set of tag names that have been used for all the posts that have an accepted answer.
6. **(3 Points)** For each year in chronological order, show the different number of posts and tags that were used.
7. **(3 Points)** Show the number of questions closed in year 2015 by month in chronological order.
8. **(3 Points)** Show the names and frequencies of the tags that are used together with the tag "intel".
9. **(3 Points)** Show the set of users who have the highest reputation and the lowest down_votes than any other user. HINT: there is no user that is better than all other users on each of the criterion individually. Thus, you need a query that can eliminate users that are worse on both criteria than some other user (in Economics your query will return what is known as the Pareto Set).

Excercise 8.2 (18 Points)

Download the database countries from: http://www.ifis.cs.tu-bs.de/webfm_send/2232. Similar to what you did in the previous exercise, use pgAdmin III to create your database. The database countries contains information about the most populous world cities as well as country level economic data, population data, and geographic data. The database also contains information on languages spoken in each country. Take your time to get a sense for the types of data that each table contains.

1. **(3 Points)** Show the name of the Central American countries with an official language for the year 2015. For each country show also the following information: total investment and imports. Order your results by country name.
2. **(3 Points)** Show the average fertility rate for each region in 2015. Order your results by average
3. **(3 Points)** Show the country code, inflation rate and unemployment rate of each country whose form of government is "Constitutional Monarchy" or some form of 'Republic'. Order your results by inflation rate. (do not use joins, use only subqueries)

4. **(3 Points)** Show the names of the cities of countries that are included in either economies or currencies but not in populations.
5. **(3 Points)** Show the percentage increase in population from 2010 to 2015 for each country code.
6. **(3 Points)** Show the country with the maximum inflation rate per continent in 2015.