Lab assignment 4

TI1506 2016/17; ti1506-ewi@tudelft.nl

Introduction

In the first part of the assignment you will reflect on the design of your application from the perspective of the data managed by it, and produce an updated application and/or database design.

Then, you will experiment more with the relational database environment by creating some queries, and testing their execution on the TODO database.

The third part of the assignment will require you to make another step towards a fully functional Web application: you will improve the node.js solution created in the previous assignment with the code required to retrieve data from the database.

The last two parts build on the database-enabled version of your to-do application. Your goal is to add analytics[1] functionalities by creating a data dashboard. Given that your application persistently stores in the database all the created/completed to-dos, your objective is to make use of the full power of SQL to provide users with insights about their usage behavior.

Deadline: when this assignment is due depends on the cluster your team is in. Every team is assessed biweekly. Make sure you know your cluster and what this means for your lab deadlines. All of this information is available in the course syllabus (on blackboard).

In order to pass the assessment, you have to make a valid attempt at working through the lab assignment.

[1] http://en.wikipedia.org/wiki/Analytics

1. Compare the Schema of the TODO Database with your Design

Critically analyze the differences existing between the provided database schema and the in-memory data structure you created during the server-side JavaScript part of Lab Assignment 3.

1. If the database schema contains more concepts (tables), attributes, and relationships than your current implementation, then think about how you should modify your application accordingly.

2. If your application provides a richer data model, propose a modification of the database by extending the drawing of **deliverable 6** of Lab Assignment 2 (the visual representation of the schema)

Deliverable: An updated version of the application design (either as a mockup or as a modified HTML template) which includes parts of the data model not currently included. Also, an updated version of the database schema (as a drawing), if your design/application included additional data. In both cases, you should be able to clearly explain the motivations behind the changes in the design/database schema.

2. Query TODO Database

Compose and execute the SQL required to provide the correct answer to the following queries.

1. List all the ToDoLists belonging to a given user. The identifier of the user should be specified as a condition in the WHERE clause of the query.

2. List all the ToDoItems belonging to a given ToDoList. The identifier of the ToDoList should be specified as a condition in the WHERE clause of the query.

3. As in (2), but now allow for pagination of ToDoItems. This means that the query should show a pre-defined amount of results, starting from a given tuple. Use the LIMIT clause implemented in MySQL (http://dev.mysql.com/doc/refman/5.7/en/select.html)

4. Add to the query in (3) the ability to filter ToDoItems according to 1) A range of date creation, 2) a priority level, 3) the completion status.

5. For a given ToDoItem, show all its sub-items.

6. For a given ToDoItem, show the value of all its tags.

7 For a given tag, show all the ToDoLists that contain ToDoItems which are tagged with that tag.

8 For each tag, calculate the number of currently pending and completed to-dos.

9 For each week in the current year, calculate the number of completed to-dos.

10 For each tag, retrieve the 10 to-dos with quickest completion time (i.e. the time between the creation and the completion of the to-do).

11 Calculate the frequency of co-occurrence of tags (i.e. the number of times each possible combination of tag pairs is used in the database)

12 For a given ToDoList, calculated the average time of completion of to-dos

13 List the to-dos having a completion time higher than the average time of completion for the to-dos belonging to the same ToDoList

Deliverable: For each of the information needs above, write your query in a text file, together with the retrieved answer list. Make sure to have the text file with you during the assessment, and be prepared to execute your queries "live" on your virtual machine.

Additional exercise: students that decided to make full use of the todo database -- i.e. by using the User and ToDoAssignment tables -- can extend all the above queries to allow filtering per a specific user.

---

3. Database Interaction with node.js

Improve the node.js script you wrote in Lab Assignment 3. Instead of keeping a list of todos in the server's memory (which only works well if the server does not crash or runs out of memory), we will now use a database to store/retrieve/update our todos.

Change your node.js script so that the server now:

1 retrieves the ToDoItems from the database upon a client's request

2 adds a new ToDoItem to the database, upon a client's submission of a new todo

3 modifies an existing ToDoItem in the database, upon a client's submission of changes related to an existing todo

4 deletes an existing todo from the database, upon a client's request for deletion

In the basic version of this exercise, the User and the ToDoList can be fixed (i.e. they are constant selection conditions in the SQL query that retrieves ToDoItems).

Deliverable: Updated server-side code of your Web application, running in your virtual machine.

Additional optional exercise: expanding your code to support multiple users and multiple ToDoLists.

---

4. Extension of Server Side Application With Analytics Functionalities

Improve your Node.js application by adding a new new *route* (i.e `app.get("/$URL$", function (req, res) { ... }`) to serve a new analytics *Dashboard* HTML page. Here is an example of dashboard page you can use as an inspiration for your dashboard "look&feel". The page contains several "widgets", each devoted to the visualization of some aggregated data. An example widget is n histogram showing the average completion time for each tag in the DB (query 10 of exercise **2**).

You will be asked to create the actual HTML page in exercise 5. In this exercise, the goal is to create the server-side functionality that will allow you to feed data to the dashboard page.

For each of the queries in **2)**, create a new *route* (i.e app.get("/URL", function (req, res) { ... that implements the logic required to execute the query on the database. Each new route must return a valid JSON response, to be later used by the *Dashboard* HTML page.

Deliverable: Updated server-side code of your Web application, running in your virtual machine.

---

5. Extension of Client-Side Application with New Analytics

Create the new *Dashboard* HTML page. The page must make use of dynamic Javascript code to:

1 Asynchronously (i.e. using AJAX requests) invoke the new endpoints created in 4)

2 Update the content of the page with the results retrieved from the server.

Visualizing data in a textual form is an acceptable solution, although the usage of visual widgets is preferred.

Deliverable: Updated client-side code of your Web application, running in your virtual machine.