StackOverflow Viewer Application

Group members

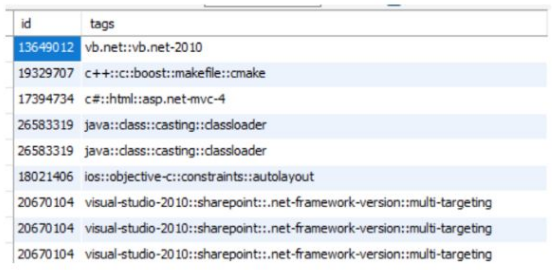
Thomas Halberg, 03528073
Søren Ulrik Johansen, 55121
Baptiste Jouan, 62147
Jean-Alexandre Pecontal, 62150
Frederik Nordam Rosenvilde Jakobsen, 55478

Introduction

The goal of this portfolio subproject 1 is to provide a database for the SOVA application (Stack Overflow Viewer Application) and to prepare the key functionality of the application.

The first part of the three part project will contain a sketch and a description of the application design with details on which features we wish to provide for the users. Furthermore the project will contain a QA-model that will represent the acquired data from Stack Overflow. But also a history and marking model that will keep track of the previous searches (search history), and support the making of, and adding notation for the search results.

Database Solutions

Atomic Values	Explanation
	The table seen on the left hand side (Table 1) shows a table with non-atomic values. These values are hard to extract because they can relate to different values. We wanted to make all values atomic by using 1st Normal Form, and therefore only make them consist of a single value. One way to do this is to split the table into two, joined on a primary key - in our case 'id' (Table 2). In our case we have connected all values in a single table joined on the primary key, where all values are atomic. (Silberschatz, Korth & Sudarshan 2011, page 329).
Table 1	

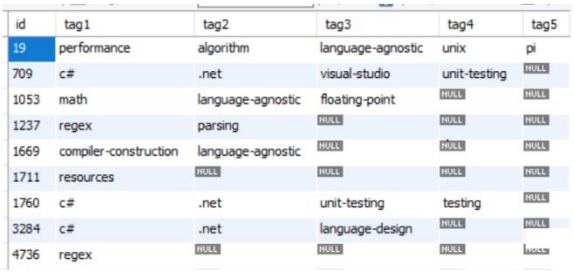



Table 2	How we did this (in steps)
drop table if exists poststag; create table poststag (id, int(11), tag1 varchar (100), int(11), tag2 varchar (100), int(11), tag3 varchar (100), int(11), tag4 varchar (100), int(11), tag5 varchar (100), primary key (id));	First of all we needed to make tables poststag.

id	tags
13649012	vb.net::vb.net-2010
19329707	c++::c::boost::makefile::cmake
17394734	c#::html::asp.net-mvc-4
26583319	java::class::casting::classloader
26583319	java::class::casting::classloader
18021406	ios::objective-c::constraints::autolayout
20670104	visual-studio-2010::sharepoint::.net-framework-version::multi-targeting
20670104	visual-studio-2010::sharepoint::.net-framework-version::multi-targeting
20670104	visual-studio-2010::sharepoint::.net-framework-version::multi-targeting

id	tag1	tag2	tag3	tag4	tag5
19	performance	algorithm	language-agnostic	unix	pi
709	c#	.net	visual-studio	unit-testing	NULL
1053	math	language-agnostic	floating-point	NULL	NULL
1237	regex	parsing	NULL	NULL	NULL
1669	compiler-construction	language-agnostic	NULL	NULL	NULL
1711	resources	NULL	NULL	NULL	NULL
1760	c#	.net	unit-testing	testing	NULL
3284	c#	.net	language-design	NULL	NULL
4736	regex	NULL	NULL	NULL	

```
insert into poststag (id,tag1,tag2,tag3,tag4,tag5)
 select id,part1 as tag1,part2 as tag2, part3 as
tag3, part4 as tag4, part5 as tag5
            from (select distinct id,
              substring_index(substring_index(tags, '::',
-5), '::',1) as part1,
    substring_index(substring_index(tags, '::', -4),
'::',1) as part2,
    substring_index(substring_index(tags, '::', -3),
'::',1) as part3,
    substring_index(substring_index(tags, '::', -2),
'::', 1)as part4,
    substring_index(tags, '::', -1) as part5
                                FROM
stackoverflow_sample_universal.posts
                                            where
posttypeid=1) as xx
       where (part1 <> part2 and part2 <> part3 and
part3 <> part4 and part4 <> part5);
```

Then we load data into table

This shows all id with 5 attribute tags is loaded into the table.

```
insert into poststag (id,tag1,tag2,tag3,tag4,tag5)
select id,part2 as tag1, part3 as tag2, part4 as
tag3, part5 as tag4, null
            from (select distinct id,
              substring_index(substring_index(tags, '::',
-5), '::',1) as part1,
    substring_index(substring_index(tags, '::', -4),
'::',1) as part2,
    substring_index(substring_index(tags, '::', -3),
'::',1) as part3,
    substring_index(substring_index(tags, '::', -2),
'::', 1)as part4,
    substring_index(tags, '::', -1) as part5
                                FROM
stackoverflow_sample_universal.posts
                                            where
posttypeid=1) as xx

       where (part2 <> part3 and part3 <> part4 and
part4 <> part5 and part1 = part2);
```

Next all id with 4 attribute tags is loaded into table, and the 5th is set to null.

```
insert into poststag (id,tag1,tag2,tag3,tag4,tag5)
select id,part3 as tag1, part4 as tag2, part5 as
tag3, null,null
            from (select distinct id,
              substring_index(substring_index(tags, '::',
-5), '::',1) as part1,
    substring_index(substring_index(tags, '::', -4),
'::',1) as part2,
    substring_index(substring_index(tags, '::', -3),
'::',1) as part3,
    substring_index(substring_index(tags, '::', -2),
'::', 1)as part4,
    substring_index(tags, '::', -1) as part5
                                FROM
```

And all id with 3 attribute tags is loaded into table, and the 4th and 5th is set to null.

```
stackoverflow_sample_universal.posts
                                    where
posttypeid=1) as xx
      where (part3 <> part4 and part4 <> part5 and
part1 = part2 and part2 = part3);
```

```
insert into poststag (id,tag1,tag2,tag3,tag4,tag5)
select id,part4 as tag1, part5 as tag2, null,null,null
          from (select distinct id,
            substring_index(substring_index(tags, '::',
-5), '::',1) as part1,
    substring_index(substring_index(tags, '::', -4),
'::',1) as part2,
    substring_index(substring_index(tags, '::', -3),
'::',1) as part3,
    substring_index(substring_index(tags, '::', -2),
'::', 1)as part4,
    substring_index(tags, '::', -1) as part5
                                FROM
stackoverflow_sample_universal.posts
                                    where
posttypeid=1) as xx
      where (part4 <> part5 and part1 = part2 and
part2 = part3 and part3 = part4);
```

Now all id with 2 attribute tags is loaded into table, and the 3,4 and 5th tag is set to null.

```
insert into poststag (id,tag1,tag2,tag3,tag4,tag5)
select id,part5 as tag1,null,null,null,null
          from (select distinct id,
            substring_index(substring_index(tags, '::',
-5), '::',1) as part1,
    substring_index(substring_index(tags, '::', -4),
'::',1) as part2,
    substring_index(substring_index(tags, '::', -3),
'::',1) as part3,
    substring_index(substring_index(tags, '::', -2),
'::', 1)as part4,
    substring_index(tags, '::', -1) as part5
                                FROM
stackoverflow_sample_universal.posts
                                    where
posttypeid=1) as xx
      where (part1 = part2 and part2 = part3 and
part3 = part4 and part4 = part5);
```

Now all id with only 1 attribute tag loaded into table, and the 2,3,4 and 5th tag is set to null.

SQL: Create Database	Explanation:
drop DATABASE if exists SOVA; CREATE DATABASE SOVA; Fig. 1: creating database. create table postsindhold (id int(11), posttypeid int(11), owneruserid int(11), parentid int(11), acceptedanswerid int(11), creationdate datetime, score int(11), body longtext, closeddate datetime, title varchar(200), primary key (id)); Fig. 2: creating tables. insert into postsindhold (id, posttypeid, owneruserid, parentid,acceptedanswerid, creationdate,score, body, closeddate, title) SELECT distinct id, posttypeid, owneruserid, parentid,acceptedanswerid, creationdate,score, body, closeddate, title FROM stackoverflow_sample_universal.posts; Fig. 3: importing data. SET SESSION sql_mode = 'STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION'; ALTER SCHEMA `SOVA` DEFAULT CHARACTER SET utf8 ; Fig. 4: configuring database.	The goal is to split the existing database, so we made new tables and we imported relevant information, as we've seen previously. An example for posts. First we create the database (fig. 1). Then we create our tables (fig.2). Finally we import data (fig.3). We also noticed that some configurations could change between computers (and mysql version), so we added few configuration lines to make sure the language is in utf8 and that we are not using full group by clause (fig. 4).

SQL: Search Procedures	Explanation:

drop procedure if exists searchposts;
delimiter //
create procedure searchposts (wordstring varchar(255))
begin
select id, posttypeid, owneruserid, parentid, acceptedanswerid, creationdate, score, body, closeddate, title, linkpostid
from posts1
where body like concat('%',wordstring,'%')
or title like concat('%',wordstring,'%');
end;//
 	
Call searchposts('Proper Variable Declaration in C');
 	
Fig. 5: research function on posts
	For the search part of our application, we have made few procedures we found interesting, as we don't yet know what data will be needed at a later stage.
	
	First thing we did was to have one basic research function on posts (parent posts or answers), comparing a string to the content of posts' title or posts' body. (fig.5)

drop procedure if exists searchcomments;
delimiter //
create procedure searchcomments (wordstring varchar(255))
begin
select id, posttypeid, owneruserid, parentid, acceptedanswerid, creationdate, score, body, closeddate, title, linkpostid, commenttext
from posts1 P, commentsbody B
where B.commenttext like concat('%',wordstring,'%')
and B.postid = P.id;
end;//
 	
Call searchcomments('need more sleep');
 	
Fig. 6: research function on comments
	Also, we created a research function on comments, comparing a string to the content of comments' body and returning the associated post.(fig.6)

drop procedure if exists search3; delimiter // create procedure search3 (wordstring varchar(255)) begin select title, questionbody,answerbody,commenttext, questionscore,answerscore,commentscore from (select p1.id as questionid,p2.parentid as answerparentid, p2.id as answerid,commentid, p1.body as questionbody, p1.posttypeid as questionposttypeid, p1.score as questionscore, p1.owneruserid as owneruserid ,p2.posttypeid as answerposttypeid,	Finally, we did a research function combining both posts and comments, returning joined rows from both tables.(fig.7)

```
      p2.body as answerbody,p2.score as
answerscore,p1.title as title ,commenttext,
commentscore
                              from postsindhold as p1,
postsindhold as p2, commentsbody
                                    where p1.id in
(p2.parentid) and p1.owneruserid =
p2.owneruserid

            and p1.owneruserid = p2.owneruserid and
p1.id = postid) as xx
  where questionbody like
concat('%%',wordstring,'%%') or answerbody like
concat('%%',wordstring,'%%')
       or commenttext like
concat('%%',wordstring,'%%') or title like
concat('%%',wordstring,'%%')
       group by questionid, commentid
            order by questionscore desc
;
end;//
Call search3('how to');
```

Fig. 7: research function on both

SQL: Stored procedures	Explanation:

``drop procedure if exists getParent;
delimiter //
create procedure getParent (childParentid int(11))
begin
select id, title, posttypeid, owneruserid,
parentid, acceptedanswerid, creationdate, score,
body, closeddate, linkpostid
from posts1
where id = childParentid
group by id;
end;//
 	
Call getParent(3279543);```
	As previously, we will create procedures to make some actions we will need in the future.
	
	We did split processes into two parts : the ones for the SOVA and the ones for the search history and marking.
	
	For SOVA, we need search functions that we already have talked about, and we also need practical functions as getting parent post from a post (fig.8), getting all child (answers) from a post (fig.9) or getting all comments from a post. (fig.10)

Fig. 8: get parent from child-post id

```
drop procedure if exists getChild;
delimiter //
create procedure getChild (idParent int(11))
begin
            select id, title, posttypeid, owneruserid,
parentid, acceptedanswerid, creationdate, score,
body, closeddate, linkpostid
                        from posts1
```

```
                    where  parentid = idParent
        group by id
   order by score desc;
   end;//

Call getChild(427217);
```

Fig. 9: get all child from parent-post id

```
drop procedure if exists getComments;
delimiter //
create procedure getComments (idPost int(11))
begin
            select commentid, postid, commentscore,
commenttext, commentcreatedate, userid
                    from commentsbody
                                where  postid = idPost
   order by commentscore desc;
   end;//

Call getComments(427217);
```

Fig. 10: get all comments from post id

```
delimiter //
create procedure searchHistoryAdd (sHid
int,sHstring varcharacter(100),sHdate datetime)
begin
            insert into searchhistory
(searchnumberid,searchstring, searchdate) values
(sHid,sHstring,sHdate);
   end;//

Call searchHistoryAdd(1,'test-title','2010-04-02
15:28:22');

select * from searchhistory;
drop procedure if exists searchHistoryRemove;
```

Fig. 11: add data to search history

```
delimiter //
create procedure searchHistoryRemove (sHid int)
begin
            delete from searchhistory
   where searchhistory.searchnumberid like sHid;
   end;//

Call searchHistoryRemove(1);
```

Fig. 12: remove data from search history

For Search history and marking, we need basics functions as adding (fig.11), erasing (fig.12) or updating data in our tables (fig.13);

```
delimiter //
create procedure personalNotesUpdate
(pNnoteid int(11),pNuserid int(11),pNpostid
int(11),pNcommentsid int(11),pNid
int(11),pNnotestring
varcharacter(100),pNnotecreationdate datetime)
begin
            update personalnotes
   set userid = pNuserid, postid = pNpostid,
commentsid = pNcommentsid, id = pNid,
notestring = pNnotestring, notecreationdate =
pNnotecreationdate
   where personalnotes.noteid like pNnoteid;
   end;//

Call
personalNotesUpdate(1,3,4,5,6,'test-update-title','2
010-04-02 15:28:22')
```

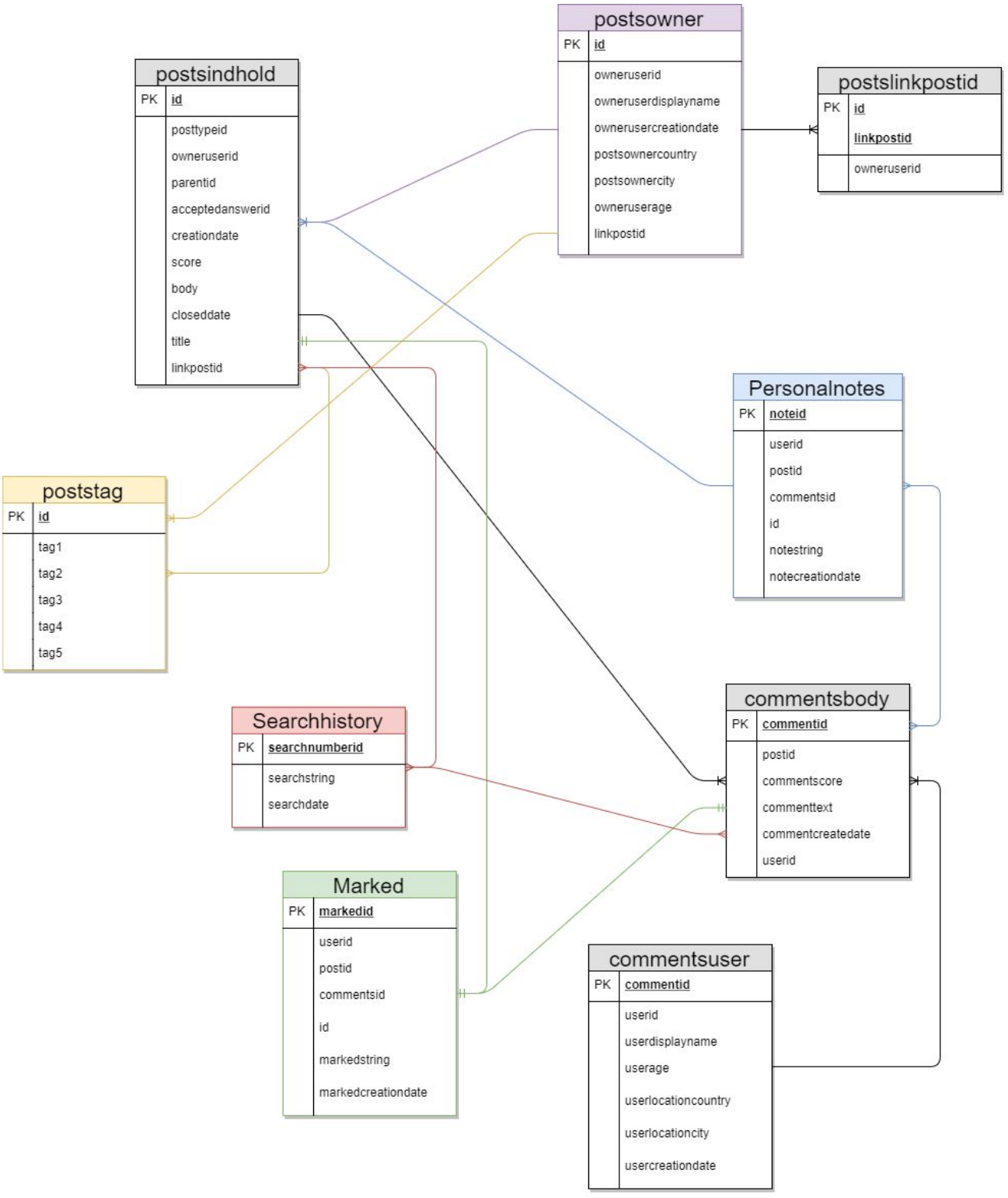
Fig. 13: update personal notes

SQL Test	Explanation
call getAnswers (427217); call getComments (427217); call getParent (3279543); call personalNotesAdd (1,2,3,4,5, 'test title', '2017-01-01 22:23:24'); call personalNotesAdd (5,6,7,8,9, 'test title2', '2017-02-01 22:23:24'); call personalNotesRemove (5,6,7,8,9, 'test title2', '2017-02-01 22:23:24'); call personalNotesUpdate (1,23,23,24,25, 'test title date', '2017-03-01 22:23:24'); call searchcomments ('need more sleep'); call searchposts ('proper variable declaration in c'); call searchHistoryAdd (1,'test-title','2017-02-22 22:23:24'); call searchHistoryAdd (2,'test-title2','2017-02-22 22:23:24'); call searchHistoryRemove (1); call search3 ('how to')	When our procedures are done, we need to test them (fig.14). Results are available in the "atachments" part of this rapport.

Fig. 14: procedures test

ER-Diagram

Diagram :



Description :

This diagram represents our database model and the relations between the tables (the colors don't mean anything, there are just there to highlight the different relations between the tables).
The main tables used in the database are the "*postsowner*" and the "*postsindhold*" tables, the former stocks the informations about each user and the later stocks all the necessary information for the posts (like the body, the score, the dates and references to the owner), we've got a relation One-to-Many because one user can post multiple times on the application and is not limited to one post, it is the same for the comments with the "*commentsuser*" and the "*commentsbody*" tables.
The only tables left are the "*poststag*" tables used to stock the different tags for the posts, there is a relation of Many-to-Many between this one and the "*postsbody*" table because one tag can be on different posts and one post can have multiple tags. The last table is "*postslinkpostid*" which serves as an intermediate table between the "*postsowner*" table and the "*postsindhold*" table. It has the same type of relation with "*postsowner*" as "postsindhold" and "postowner".
As for the tables "*Marked*", "*Personalnotes*" and "*Searchhistory*" they are all linked to "*postindhold*" and "*commentsbody*" with Many-to-Many relations except for the "*Marked*" table which has a One-to-Many relation with both tables because you can only mark one post/comment at a time, the post/comment has a binary state; marked or unmarked.

Stackoverflow User Application

In this section we will describe the requirements and functionality we have used for our graphical user interface (GUI). Note that some design or functionality could change later as it's a basic sketch of what we want to achieve.

Minimum requirements

- Search for posts and comment in Stack Overflow.
- Present search results as (ranked) list of posts.
- Visualize search results by most frequent words using ranked list or word cloud.
- Keep track of search history.
- Provide a marking option for posts of special interest among posts presented in the search result and allow optional annotation to marked posts.

Extra features and functionality

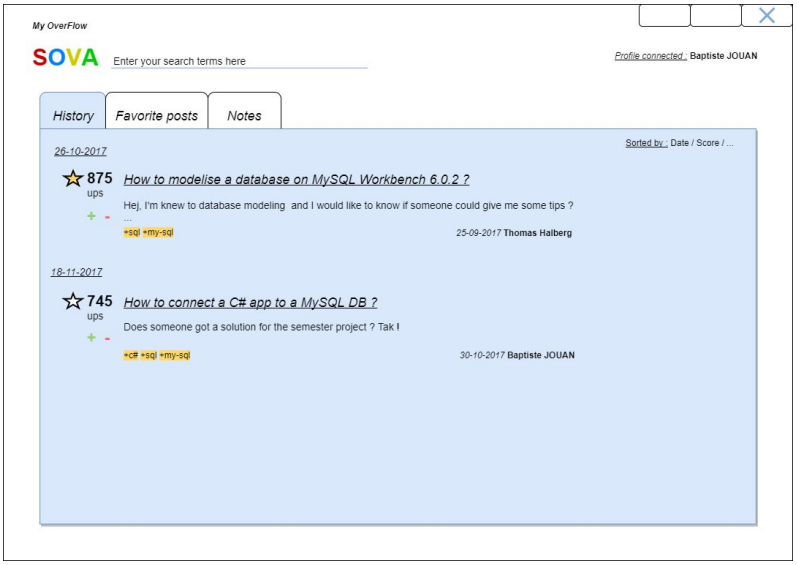
- Profile options
    - login
    - connection status
- Phrases and words search

Present the GUI

Actions valid for the entire application

Across the different iterations of the application below, you'll see that some actions stay the same through the screens.
Whenever you click on a post link it will send you to the page related to this post (cf Fifth iteration).
If you click on the star on the left of a post you can add/delete it from your favourites and you'll be able to see them in the page dedicated to it (cf Second iteration).
Also the application provide the user with a function for searching in 'comments' and 'posts' wherever he/she is.

First iteration GUI: Search History, Present, Profile.

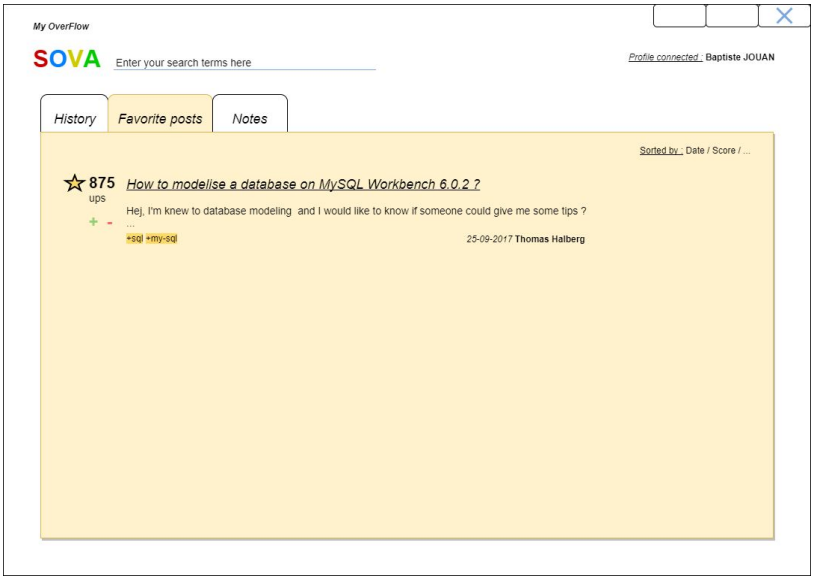


The search history will be shown in the *History* section below the search bar. In this iteration the search results, shown in *History*, is listed as a ranked list descending from the posts or comments with the highest rating.

For further iteration we have considered adding different ways of ranking the search history. We have considered following ways of ranking the results descending from.

- Posts with the highest number of comments.
- The newest or oldest post.
- Post descending from date/time.
- Post with a specific owner/user id or display name.
- Posts with specific tags.

Second iteration GUI: Favorite posts (Star marked posts)

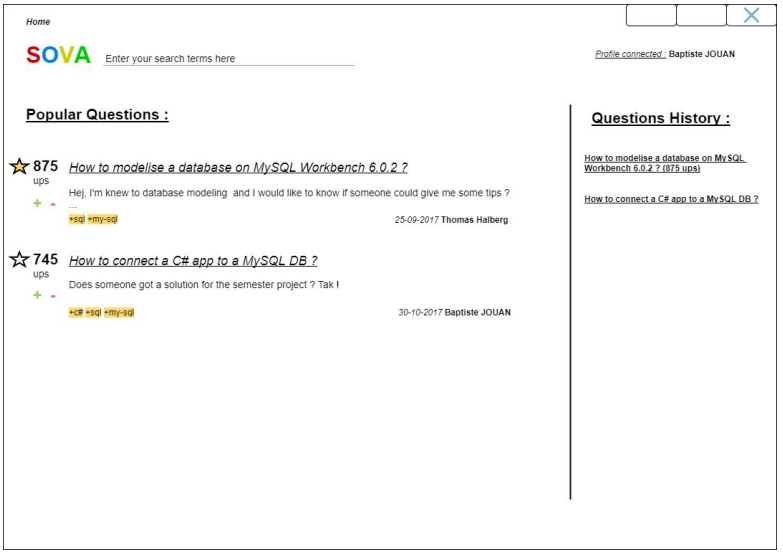


The second iteration of the GUI include a side where you can preview all your saved favourite posts. By clicking on the post (in favourite posts) the user will return to the fifth iteration and will be able to see the thread of comments to the post.

Third iteration GUI: Annotations for a specific posts

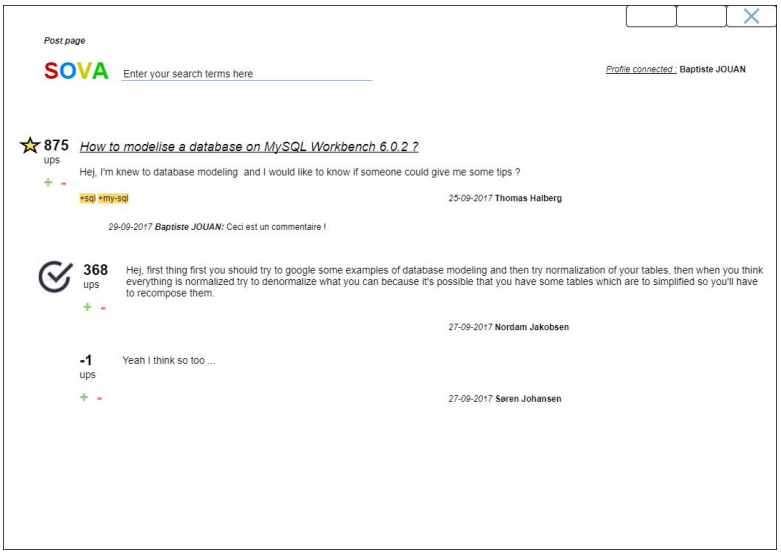
This iteration is used to visualize the different answers and comments you could have posted under a post, like the last two iterations you can order the answers/comments by dates, scores, etc ...

Fourth iteration GUI: Popular Questions, Search History



This is the Home page of the application, you will begin here. When you arrive the first thing that will be displayed are the popular questions, they are ordered by scores, you also got a minimal history of the questions you already saw.
This part of the GUI will also shows the search result. These search results is based on the strings and tags given in the search field.

Fifth iteration GUI: Search Result



This iteration represents the page where you can see the whole post, you can see the original question, the answers, the comments, their scores, etc ...
The answer section is ranked by amount of upvotes a post have been given. The owner of the initial question can also certified one of the answers as to attest that the answers helped him/her.

Bibliography

Silberschatz, Korth & Sudarshan 2011, *Database System Concepts* Sixth Edition, McGraw-Hill Education.

Atachments

Results of SQL tests :

Output:

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	16:27:23	use raw4	0 row(s) affected	0.000 sec
2	16:27:23	call getAnswers (427217)	4 row(s) returned	0.016 sec / 0.000 sec
3	16:27:23	call getComments (427217)	3 row(s) returned	0.031 sec / 0.000 sec
4	16:27:23	call getParent (3279543)	1 row(s) returned	0.000 sec / 0.000 sec
5	16:27:23	call personalNotesAdd (1,2,3,4,5, 'test title', '2017-01-01 22:23:24')	1 row(s) affected	0.000 sec
6	16:27:23	call personalNotesAdd (5,6,7,8,9, 'test title2', '2017-02-01 22:23:24')	1 row(s) affected	0.000 sec
7	16:27:23	call personalNotesRemove (5)	1 row(s) affected	0.000 sec
8	16:27:23	call personalNotesUpdate (1,23,23,24,25, 'test title date', '2017-03-01 22:23:24')	1 row(s) affected	0.000 sec
9	16:27:23	call searchcomments ('need more sleep')	1 row(s) returned	0.110 sec / 0.000 sec
10	16:27:23	call searchposts ('proper variable declaration in c')	1 row(s) returned	0.125 sec / 0.000 sec
11	16:27:23	call searchHistoryAdd (1,'we are trying til learn','2017-02-22 22:23:24')	1 row(s) affected	0.000 sec
12	16:27:23	call searchHistoryAdd (2,'we are trying til learn 2','2017-02-22 22:23:24')	1 row(s) affected	0.015 sec
13	16:27:23	call searchHistoryRemove (2)	1 row(s) affected	0.000 sec
14	16:27:23	call search3 ('how to')	224 row(s) returned	2.094 sec / 0.000 sec

Get answers:

id	title	posttypeid	owneruserid	parentid	acceptedanswerid	creationdate	score	body	closeddate
427231	NULL	2	3	427217	NULL	2009-01-09 06:20:25	19	<p>Your second example won't compile, as the...	NULL
427237	NULL	2	22656	427217	NULL	2009-01-09 06:24:25	6	<p>As others have mentioned, your second ex...	NULL
427238	NULL	2	37494	427217	NULL	2009-01-09 06:25:40	1	<p><a href="http://www.codinghorror.com/bl...	NULL
427257	NULL	2	37020	427217	NULL	2009-01-09 06:38:35	-1	<p>Having a trivial getter and setter is very si...	NULL

Get comments :

commentid	postid	commentscore	commenttext	commentcreatedate	userid
250970	427217	1	The second example still wouldn't compile - you ...	2009-01-09 07:27:34	22656
250940	427217	0	doh, you're right... typo. need more sleep or c...	2009-01-09 07:07:40	51949
250853	427217	0	"Isn't it just as effective in using a public variabl...	2009-01-09 06:16:10	16076

Get parent :

id	title	posttypeid	owneruserid	parentid	acceptedanswerid	creationdate	score	body	closeddate
3279543	What is the copy-and-swap idiom?	1	87234	NULL	3279550	2010-07-19 08:42:10	792	<p>What is this idiom and when should it be us...	NULL

Search in comments:

id	posttypeid	owneruserid	parentid	acceptedanswerid	creationdate	score	body	closeddate	title	commenttext
427217	1	51949	NULL	427238	2009-01-09 06:11:30	2	<p>I noticed some people decl...	NULL	Proper Variable Declaration in C#	doh, you're right... typo. need more sleep or c...

Search in posts :

id	posttypeid	owneruserid	parentid	acceptedanswerid	creationdate	score	body	closeddate	title
427217	1	51949	NULL	427238	2009-01-09 06:11:30	2	<p>I noticed some people declare a private var...	NULL	Proper Variable Declaration in C#

Search in both :

title	questionbody	answerbody	commenttext	questionscore	answerscore	commentscore
Why not use tables for layout in ...	It seems to be the <a href="http://stacko...	Funny:

 <a href="ht... | @Camilo SO still lives in the 20th century. Jeff a... | 665 | 25 | 1 |
What are your favorite extensio...	Let's make a list of answers where you pos...	The ThrowIfArgumentIsNull is a nice way t...	The codeplex page is empty...	478	23	0
What are your favorite extensio...	Let's make a list of answers where you pos...	The ThrowIfArgumentIsNull is a nice way t...	count me in... I am on codeplex as chakrit. sam...	478	23	0
What are your favorite extensio...	Let's make a list of answers where you pos...	The ThrowIfArgumentIsNull is a nice way t...	I'm still here yes!!! ;-) Just not as active as I wo...	478	23	0
What are your favorite extensio...	Let's make a list of answers where you pos...	The ThrowIfArgumentIsNull is a nice way t...	There is also Mono.Rocks which provides many ...	478	23	0
What are your favorite extensio...	Let's make a list of answers where you pos...	The ThrowIfArgumentIsNull is a nice way t...	I'd be happy to join on codeplex! Name over th...	478	23	0
What are your favorite extensio...	Let's make a list of answers where you pos...	The ThrowIfArgumentIsNull is a nice way t...	Is this still active? I'd like to join the fun	478	23	0
What are your favorite extensio...	Let's make a list of answers where you pos...	The ThrowIfArgumentIsNull is a nice way t...	really... Google Code pretty please?	478	23	0
What are your favorite extensio...	Let's make a list of answers where you pos...	The ThrowIfArgumentIsNull is a nice way t...	I know... but I doubt it'll be as solid as Google's ...	478	23	1
What are your favorite extensio...	Let's make a list of answers where you pos...	The ThrowIfArgumentIsNull is a nice way t...	Btw, perhaps, shouldn't the project have a pro...	478	23	0
What are your favorite extensio...	Let's make a list of answers where you pos...	The ThrowIfArgumentIsNull is a nice way t...	If possible, I'd vote for Google Code instead of ...	478	23	7
What are your favorite extensio...	Let's make a list of answers where you pos...	The ThrowIfArgumentIsNull is a nice way t...	Now the first code is committed to the Codeplex...	478	23	0
What are your favorite extensio...	Let's make a list of answers where you pos...	The ThrowIfArgumentIsNull is a nice way t...	You have subversion support on Codeplex.	478	23	15
What are your favorite extensio...	Let's make a list of answers where you pos...	The ThrowIfArgumentIsNull is a nice way t...	Extension method madness ...	478	23	0