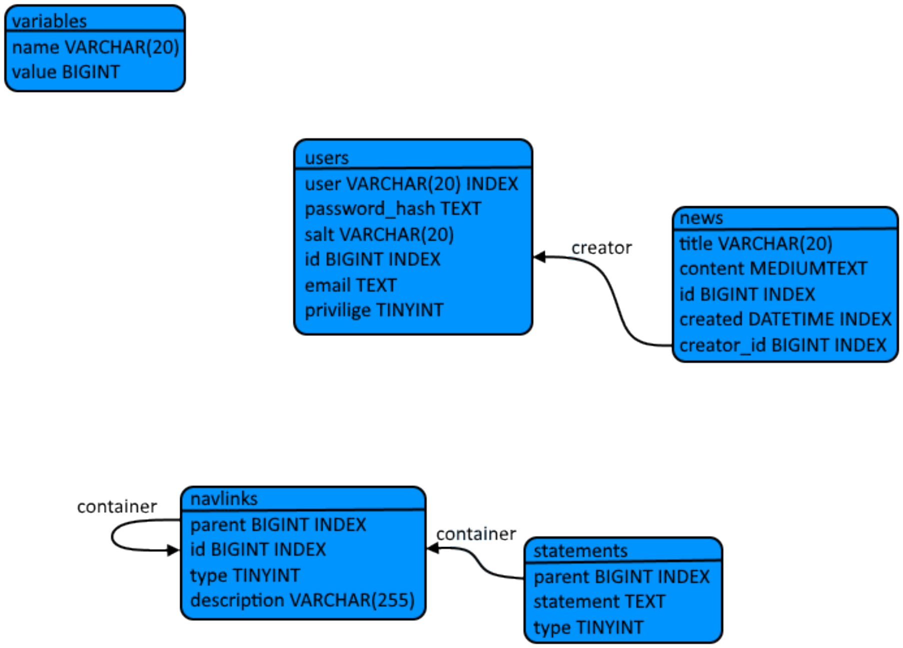
Database for EIOtraining

Oliver-Matis Lill

June 24, 2017

My goal is to make a database for all the functionality in the EIOtraining website. The database needs to store the data of the EIOtraining users, the news and the problemset entries. It's users will be mainly problem creators who create programming problems, and students who solve them. This database is necessary for the EIOtraining website to work, because users, news and prolemset entries need to be stored somewhere.

The database class model looks like this:

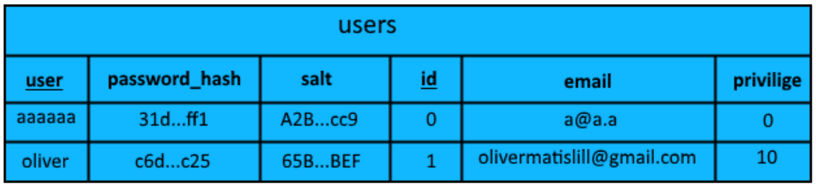


I used "paint.net" to draw this model. The usage of entities should be self explainatory. "users" contains the account data for each user, "news" contains the news with a reference to the user that created it. "variables" are simply some presistent variables used in EIOtraining. ”navlinks” is the least obvious, each navlink is basically a folder, that contains other navlinks or a statement, depending on type. ”statements” simply contains the problem/tutorial statements.

The SQL statements to create this relational database are:

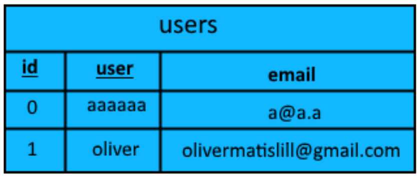
```
CREATE TABLE users (user VARCHAR(20), INDEX(user), password_hash TEXT,
      salt VARCHAR(20), id BIGINT, INDEX(id), email TEXT, privilige TINYINT);
CREATE TABLE variables (name VARCHAR(20), value BIGINT);
CREATE TABLE news (title VARCHAR(255), content MEDIUMTEXT, id BIGINT,
      INDEX(id), created DATETIME, INDEX(created), creator_id BIGINT, INDEX(
      creator_id) );
CREATE TABLE navlinks (parent BIGINT, INDEX(parent), id BIGINT, INDEX(id),
      type TINYINT, description VARCHAR(255));
CREATE TABLE statements (parent BIGINT, INDEX(parent), statement TEXT, type
      TINYINT);
```

And the ”users” table in it looks like this:

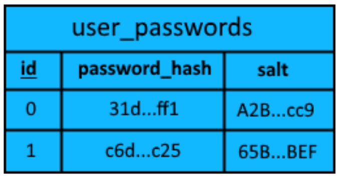


users					
user	password_hash	salt	id	email	privilige
aaaaaa	31d...ff1	A2B...cc9	0	a@a.a	0
oliver	c6d...c25	65B...BEF	1	olivermatislill@gmail.com	10

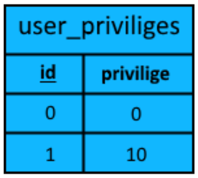
This table is obviously not in third normal form, because all this data is needed for every user anyways. Using a single table makes database smaller by avoiding duplicate attributes and indices and also makes queries much simpler. The normal form of this table would look like this:



users		
id	user	email
0	aaaaaa	a@a.a
1	oliver	olivermatislill@gmail.com



user_passwords		
id	password_hash	salt
0	31d...ff1	A2B...cc9
1	c6d...c25	65B...BEF



user_priviliges	
id	privilige
0	0
1	10

The database has been realized in the EIOtraining website using MySQL. This procedure signs up an user:

```
CREATE PROCEDURE signup (user VARCHAR(20), password_hash TEXT, salt
      VARCHAR(20), id BIGINT, email TEXT)
BEGIN
   INSERT INTO users VALUES(user, password_hash, salt, id, email, 0);
END
```

The output was:

```
Query OK, 0 rows affected (0.00 sec)
```

This procedure retrieves the data of a news story and it’s creator user:

```
SELECT ∗ FROM news INNER JOIN users ON news.creator_id = users.id WHERE
      user = ’oliver’ ORDER BY created DESC;
```

The output was:

```
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
| title | content | id | created | creator_id | user | password_hash | salt | id | email | privilige |
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
| Second Posts | This is a test on multiple posts | 5 | 2017−06−11 22:57:25 | 1 | oliver |
      c6d7abe556bdfd6e6a1f375cc05f55e90c1a7b08ce258c66bd51faf3f62d2c25 | 65
      B86d0DdCa8Bc17cBEF | 1 | oliver@omail.oml | 10 |
| First Post | Hello! This is a tes on whether the posting system works!
<marquee>HTMLinjection</marquee> | 4 | 2017−06−11 22:55:07 | 1 | oliver |
      c6d7abe556bdfd6e6a1f375cc05f55e90c1a7b08ce258c66bd51faf3f62d2c25 | 65
      B86d0DdCa8Bc17cBEF | 1 | oliver@omail.oml | 10 |
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
```

This procedure updates a statement:

```
UPDATE statements SET statement = ”Hello␣World!” WHERE parent = 3001;
```

The output was:

```
Query OK, 1 row affected (0.30 sec)
Rows matched: 1 Changed: 1 Warnings: 0
```

This procedure retrieves a statement by it’s navlink description¿

```
SELECT statement FROM statements WHERE parent = (SELECT id FROM
      navlinks WHERE description = ”Getting␣Started”);
```

The output was:

```
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
| statement |
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
| <h3>Hello!</h3>
We hope you enjoy the EIOtraining problemset! |
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
1 row in set (0.00 sec)
```

This procedure retrieves all navlinks and their statements, if present:

```
SELECT description, statement
FROM navlinks
LEFT JOIN statements ON statements.parent = navlinks.id;
```

The output was:

```
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
| description | statement |
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
| Getting Started | <h3>Hello!</h3>
We hope you enjoy the EIOtraining problemset! |
| Hello World! | Output
<marquee>Hello World!</marquee> |
| Codeforces | Hello World! |
| Eio | |
| Demoproblem | This is a demo text. |
| Introduction | NULL |
| Additional References | NULL |
| DemoFolder | NULL |
-------------------------------------------
8 rows in set (0.00 sec)
```