Getting Started with Zumero

Table of Contents

1. About this guide

This document is a brief hands-on introduction to Zumero. It shows how to get started by using the SQLite command-line shell[1] from a desktop computer.

> Right now you might be thinking, "On a desktop computer? I thought Zumero was for mobile?"
>
> You're quite right, of course. Zumero is all about mobile. But very few so-called mobile apps are accessed *exclusively* from mobile devices. You may (and probably will) want the ability to also access your data from web apps, desktop applications, or even other servers.
>
> Even if you don't need any of these things, the SQLite command-line shell is a handy way to perform administrative tasks for your Zumero server, such as initial setup of your schema, configuring permissions, or analyzing logs.

This guide does not attempt to cover all the things you can do using the SQL language with SQLite and Zumero. See the *SQLite Development with Zumero* guide for more.

This guide also does not discuss platform-specific issues such as "how to use Zumero with Objective C and Xcode", or "how to use Zumero from Java on Android". These things are covered in separate documentation.

2. What is Zumero?

To describe Zumero, we first describe SQLite.

SQLite[2] is lightweight (but surprisingly powerful) implementation of a SQL database.

SQLite is the standard database software for iOS, Android, and Windows RT. It is installed on over a billion mobile devices.

[1] http://sqlite.org/sqlite.html
[2] http://www.sqlite.org/

But like any other computer, and perhaps more so, a mobile device is not isolated. It needs to share data with a server.

And SQLite has no synchronization capabilities.

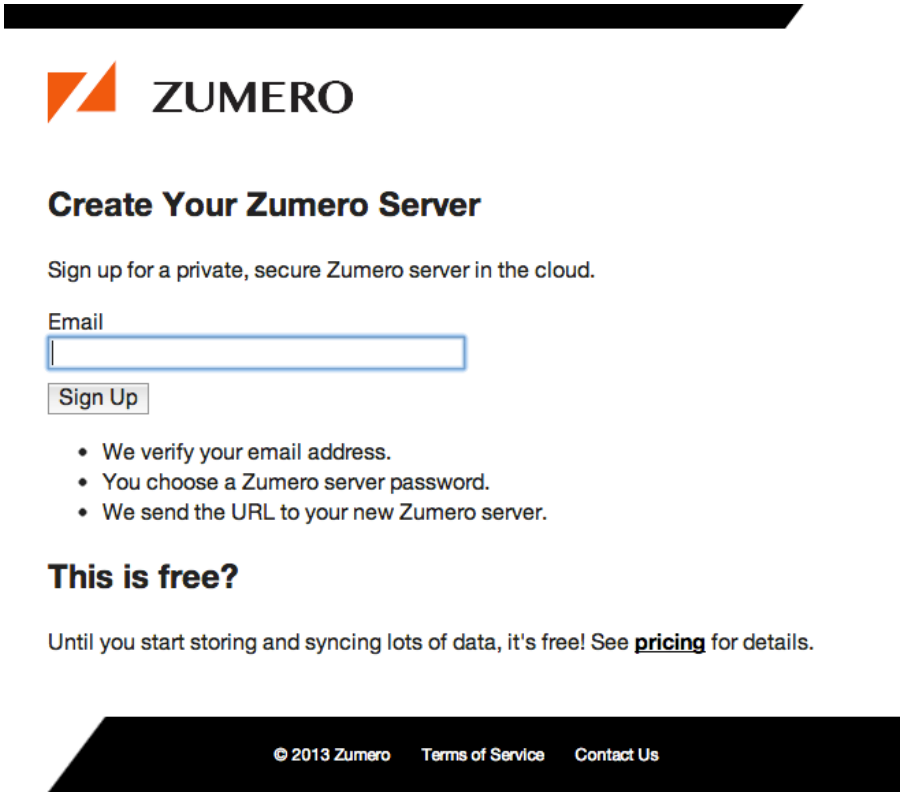
Zumero solves that problem.

Zumero is "sync for SQLite".

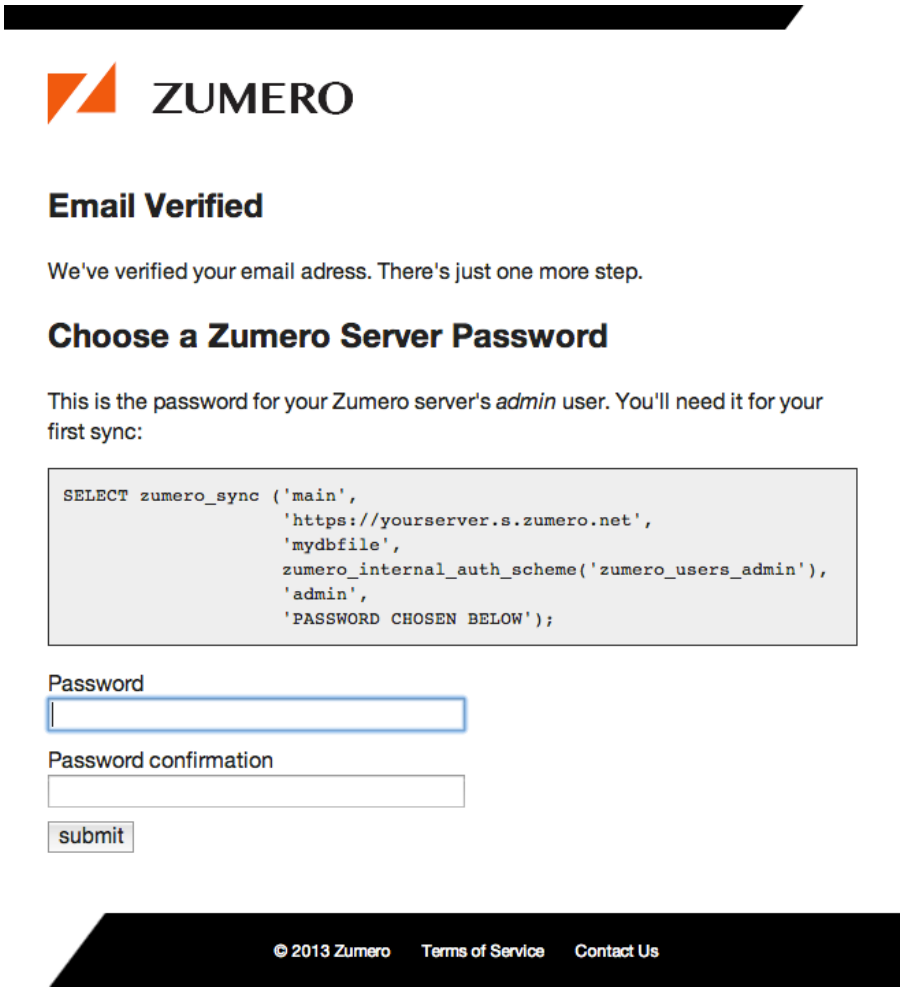
3. Sign up and create a cloud server

First you need a Zumero server. Signup is easy. No credit card required. All we need from you is your email address and an 'admin' password.

Visit https://zumero.net and enter your email address:



Very shortly you should receive a welcome email containing a link. Click the link to confirm that your email address is valid, and to finish setting up your Zumero server. You should see a web page, asking you to provide a password for your 'admin' user:



Enter a password (please make it a good one) and write it down somewhere safe. You will need it later.

After you submit the form, you should receive another email containing the URL for your Zumero server.

Your Zumero server is ready for use!

4. Check your 'sqlite3' command-line shell

Before we go any further, let's make sure you have a SQLite command-line shell application which is compatible with Zumero.

For compatibility with Zumero, you need a 'sqlite3' command-line shell which...

- ... is version 3.7.11 or higher
- ... contains support for the .load command

Most Windows computers do not have SQLite preinstalled.

Most Mac OS computers ship with SQLite preinstalled, but it was compiled with dynamic extension loading disabled.

It is therefore likely that you need to visit http://sqlite.org/download.html and download the command-line shell for your platform.

5. Load the Zumero extension

To use Zumero from the SQLite command-line shell, you first need to dynamically load the Zumero SQLite extension. This is provided as a dynamic library (.DLL, .dylib, .so, etc.) for use with the ".load" command.

In the following example, we are on MacOS, so the Zumero extension is called "zumero.dylib", and it is in the current directory:

```
$ sqlite3
SQLite version 3.7.15.2 2013-01-09 11:53:05
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> .load zumero.dylib
sqlite>
```

Note that when .load succeeds, it says nothing. On the other hand, if the .load command cannot find the library, you will get something like this:

```
sqlite> .load wrong/path/zumero.dylib
Error: dlopen(wrong/path/zumero.dylib, 10): image not found
sqlite>
```

Now that the Zumero extension is loaded, we can access Zumero features using SQL commands.

6. Accessing your server

For your security, your new Zumero server was initially configured with a big, heavy front door that is locked to keep out intruders.

When you signed up and created your server at zumero.net, you were asked to provide a password. The security settings for your new server were then configured as follows:

- Create an "internal auth" database (kind of like a password file) called "zumero_users_admin".
- Add an entry to "zumero_users_admin" with the username "admin" and the password you provided.
- Configure the server such that only members of "zumero_users_admin" can create dbfiles.

In other words, until you add new users or permissions, the user named "admin", located in the "zumero_users_admin" dbfile, is the only user who is allowed to do *anything* with your new server. You should think of your "admin" user like "root" on a Unix system, or an Administrator account for Windows. It is mostly used for administration, initial setup, and to create other users that will be endowed with fewer privileges.

For now, let's make sure you have access to your server using the password you provided at signup. To do that, we'll need to call one of the Zumero functions that actually contacts the server over the network.

But there aren't very many of them. With Zumero, very few operations actually require access to the server. The whole point of the "replicate and sync" approach is to allow the app (and its developer) to ignore issues of networking for regular database operations.

The primary Zumero function that uses the network is zumero_sync(), but you don't have anything to synchronize yet.

So, for the purpose of verifying access to your server, we'll do something simpler and call zumero_get_storage_usage_on_server(). This function is used to ask the server for a list of dbfiles, and for each one, its size in bytes.

Here is an example of a call to that function:

```
sqlite> SELECT zumero_get_storage_usage_on_server(
    'https://zinsta501bf8d197.s.zumero.net',
    zumero_internal_auth_scheme('zumero_users_admin'),
    'admin',
    '(put your password here)',
    'my_dbfile_list'
    );

sqlite>
```

The zumero_get_storage_usage_on_server() function needs 5 parameters:

- The first parameter is the URL of your server. This was provided to you in an email during the signup process.
- The next three parameters are your credentials for accessing the server. All Zumero functions which access the server will require you to specify the same three parameters as credentials:
    - The "authentication scheme string" which defines *where* your username should be authenticated. For an internal auth dbfile like "zumero_users_admin", we just use the zumero_internal_auth_scheme() function to construct the string we need.
    - The username.
    - The password.
- The final parameter is the name of a temporary table which will be created to contain the list you are requesting.

Note that we are using a SELECT statement to call this function because that is the SQL language construct for evaluating an expression. This may seem odd to you, since SELECT is usually used to retrieve rows from a table or join. But it is perfectly legal SQL to use SELECT to evaluate any expression. For example, the following statement returns an integer that simulates the rolling of a six-sided die:

```
sqlite> SELECT 1 + abs(random() % 6);
```

Note also that in our example, the actual function return value for zumero_get_storage_usage_on_server() is not of any interest. It always returns NULL when it succeeds.

The real information "returned" by zumero_get_storage_usage_on_server() is placed in the temporary table of the name you provided (my_dbfile_list, in the example above). We can get that information by querying that table:

```
sqlite> .headers ON
sqlite> SELECT * FROM my_dbfile_list;
```

Here you can see the result of zumero_get_storage_usage_on_server() is a temporary table with two columns: "name" (which contains the name of the dbfile) and "size" (which contains the disk size of that file in bytes). Since your server is newly installed, you have only three dbfiles: "zumero_users_admin" (the auth db we mentioned earlier), "zumero_config" (which contains server configuration parameters), and "zumero_log" (which contains a log of all accesses to your server).

Finally, let's come back to where we started with this section and show that it is not possible to call zumero_get_storage_usage_on_server() without providing the correct authentication credentials.

In this example, we provide the wrong username:

```
sqlite> SELECT zumero_get_storage_usage_on_server(
    'https://zinsta501bf8d197.s.zumero.net',
    zumero_internal_auth_scheme('zumero_users_admin'),
    'administrator',
    '(put your password here)',
    'my_dbfile_list'
    );
Error: zumero:authentication_failed
sqlite>
```

In this example, we provide no credentials at all:

```
sqlite> SELECT zumero_get_storage_usage_on_server(
    'https://zinsta501bf8d197.s.zumero.net',
    NULL,
    NULL,
    NULL,
    'my_dbfile_list'
    );
Error: zumero:permission_denied
sqlite>
```

7. Creating a Zumero table

The SQL statement for creating a Zumero table is nearly identical to the one for creating a regular table:

```
CREATE VIRTUAL TABLE foo USING zumero (nam TEXT UNIQUE NOT NULL, val TEXT NOT NULL);
```

Except for the fact that it can be synchronized, a Zumero table works basically just like a regular SQLite table. You can SELECT, INSERT, UPDATE and DELETE, exactly like you normally would.

All the differences between Zumero tables and regular tables are explained in the *SQLite Development with Zumero* guide, but one thing is important enough to mention here:

If you want to create an index on a Zumero table, prefix the table name with "z$":

```
CREATE INDEX ndx_foo_val ON z$foo (val);
```

8. Synchronizing

Finally, we arrive at the feature that is central to Zumero: synchronization.

To synchronize a SQLite database file with its counterpart on the server, we use the zumero_sync() function. For example:

```
sqlite> SELECT zumero_sync(
    'main',
    'https://zinsta501bf8d197.s.zumero.net',
    'dbfile_name',
    zumero_internal_auth_scheme('zumero_users_admin'),
    'admin',
    '(put your password here)'
    );
0
sqlite>
```

Here we are calling the zumero_sync() function with six parameters. The first one is the name of the attached SQLite database which should be synchronized (usually 'main'). The second parameter is the URL of the server. The last three are the authentication credentials. These are identical to what we did when we called zumero_get_storage_usage_on_server() previously.

The third parameter is the name of the dbfile on the server with which we want to synchronize. A dbfile name must begin with a lowercase letter and must contain only lowercase letters, digits, or underscores.

If the named dbfile does not yet exist, and if the credentials you provide correspond to a user who has permission to create a dbfile, then it will be created, and all Zumero tables in your local SQLite file will be synchronized up to the dbfile on the server..

In the more common case where the database already exists on the server and may in fact have changed, zumero_sync() copies incremental changes in both directions to make both copies of the dbfile match.

9. Next Steps

- Want more details? See *SQLite Development with Zumero* for much more complete coverage of Zumero's concepts, operations, and API.
- Developing for iOS? The iOS area of the Zumero Client SDK has stuff which should seem friendlier to those who spend their time with Objective C and Xcode.
- Developing for Android? The Zumero Client SDK provides a Java API which is compatible with `android.database.sqlite`.
- Got a question? You can email support@zumero.com. Or post a question on http://stackoverflow.com/ and tag it "zumero". We monitor and respond to both.