How To PouchDB and CouchDB - Database Per User Made Simple

https://titrias.com/how-to-pouchdb-couchdb-database-per-user-made-simple/

CouchDB is one of the most powerful DBMS nowadays. However, although its documentation is good, there is not enough topics of how-tos and best practices for common use cases.
One of these use cases is the database_per_user. Each user has his own private data that only that user can read or write.
I have been searching for a technique to implement a more complex application. I wanted to create an **offline-ready** system with **remote database synchronization** where some of the data are **mutual** between **all users** and some of the data are **private** for **each user**. The system should have an **analytical admin panel** with graphs and stuff like that for **the whole thing**.
This is the part where everything became very hard to implement for someone with SQL & MongoDB background like me. CouchDB has neither tables nor schemas. Just single key/value table per database where every thing can fit.
Implementing this in SQL based DBMS would be quite easy

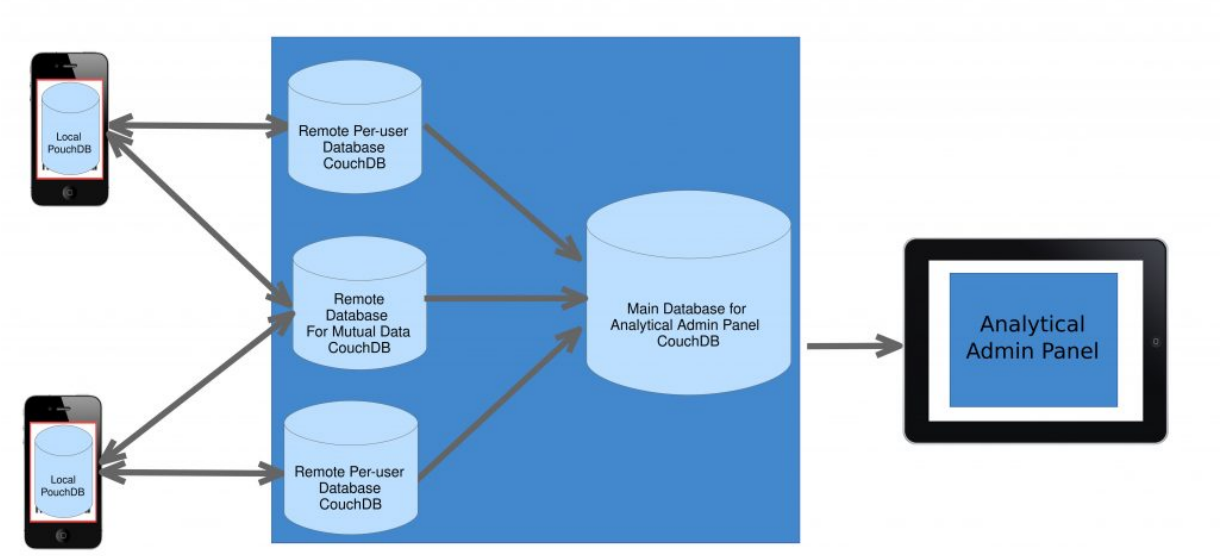
1. Create users table.
2. Create table per entity
3. Create an application tier between the database and the frontend (PHP, Node, …)
4. Handle authentication and authorization in the application tier.

Simple Solutions that will NOT work using couchDB includes :

1. Create an application tier, You will lose CouchDB native RESTful endpoints. Building replication endpoint by yourself would be a headache.
2. Create a single database and handle authentication in the client-side. That's really bad! It would be easily hackable. The client-side code – even if compiled – can be easily inspected to get the DB credentials.
3.

Many online topics about couchDB are talking about how cheap databases are. Every time you need a new database create one. Even if you needed hundred of thousands of databases. Couch can scale and replicate. So

whenever you need a database, just create one.
That concept totally changed how I think about CouchDB. I thought of the following solution:
Single database per mutual data and one database_per_user for private data. Both of them replicate to the main database in a one-way replication. Like shown in the image below.



Let's see how to create each of these components step by step.

1. Install couchDb on a server or localhost and enable CORS
2. Install CouchPerUser
    1. Follow this : https://github.com/etrepum/couchperuser
    2. On Archlinux I found the path to couchdb to be /usr/lib/couchdb/
    3. So on Arch the path to the plugin should be : /usr/lib/couchdb/plugins/couchperuser
3. Restart CouchDB
4. Now, whenever you create a new user, a new database will be automatically created for that user.
    - HINT : CouchDB admin panel : http://localhost:5984/_utils/ (replace localhost with your domain name or IP)
5. Create the mutual remote database, let's call it main-entities and create one database for the analytical dashboard to which every user database should replicate.

6. Now on the client-side :
    1. Create new PouchDB project (You can use Angular-pouch for seamless integration with angular)
    2. You should create 2 pouch databases
        1. One for the mutual data and should sync with main-entities
        2. One private database that should sync with the private user database.
7. On creating new users, enable continous integration between userdb and main database.

Let's recap all databases

Remote databases :

1. Mutual data database (Only One) (All users can read / write to this database.)
2. User database (1 per user) (Only the owner can read / write to this database)
3. Main database (Only one) (For the analytical dashboard) (NO DATA SHOULD BE WRITTEN HERE BY ANY USER, ONLY REPLICATED DATA BY CONTINOUS REPLICATION)

Local Databases : Mutual db + User db.

Is that a perfect solution ?

Ofcource not! The main disadvantage of this solution is the absense of a central offline database. The only central database here is the main database on the server. My application didn't require this. However, databases are very cheap. You can create a local central database to replicate all data to. So that you can generate offline reports if you wanted to.

Let's see some code snippets

Connect to remote database and sync :

?

```
dbs.remote.private = pouchDB(DATABASE.URL +
"userdb-" + _convertToHex(username), {
    auth: {
        username: username,
        password: password
    },
    skip_setup: true
});
 dbs.local.private.sync(dbs.remote.private).on('comp
lete', function() {

 }).on('error', function(err) {
 });
```

_convertToHex is a function responsible of converting strings to hex representation. Why ? because couch_peruser uses hex-based usernames when creating databases.

So, for example, creating a user named “titrias” by setting `_id` to `org.couchdb.user:titrias` and doc.name to `titrias` should create a new database called `userdb-74697472696173` through the couch_peruser plugin

?

```
function _convertToHex(str) {
    var hex = '';
    for (var i = 0; i < str.length; i++) {
        hex += '' + str.charCodeAt(i).toString(16);
    }
    return hex;
}
```

User Doc :

?

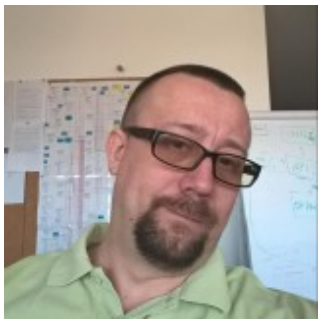
```
{
    "_id": "org.couchdb.user:titrias",
    "_rev": "1-9cc65b3e62e0210fdbc89bf6390e3be1",
    "password_scheme": "xxx",
    "iterations": 10,
    "username": "titrias",
    "type": "user",
    "name": "titrias",
    "roles": [
    ],
    "derived_key": "xxx",
    "salt": "xxx"
}
```

If there is anything still not clear, Leave it in the comments. I will try my best to add more code snippets soon.

Fady S. Ghatas

PSM™, TiTrias Founder and CEO, white hat hacker acknowledged by Microsoft, Apple, Redhat & AT&T. Publisher of GANKIN in SN Applied Sciences. Drafter at historydraft.com.

19 Comments

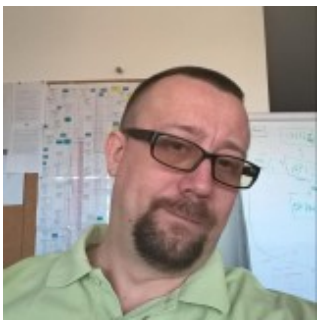


Mirza Abazovic on December 13, 2016 at 11:12 pm
Nice article. I have question about mutal data for scenario when user A wants to share data only with user B and not with others (user C, user D etc …).
Folowing this pattern solution is to create mutal database only for users A and B and that is now 3 local PouchDb for sync. This could keep going share data only for users A, C and D
Is maybe solution for this filtered replication (on server side) with one local puchdb and one CouchDB database (server) but with documents that have some obligatory data for example { owner: “user A”, readers: [“user B”,”user C], writers [“user D”]}.
Some kind of ACL contained in every doc that would be used in filtered replication to sync data to right pouchDb and also Validate Document Update Function.
What is Your opinion ? Is this idea possible to implement ?
Reply

Fady S. Ghatas on December 22, 2016 at 8:46 pm
Thanks for stopping by.

The couchdb way would be to just have a new separate database on the server side where you can use validate_doc_update to strict writing to this database to specific users. You can go one further step and add only allowed users to permissions table of the database to allow the specified users to read/write and forbid other non-administrator users from accessing this database.
Another way would be to use role where, for example, userA & userB are in HR team so there would be a role named “HR” and another for sales team and so on. So you can permit access to roles instead of users. Again this would require you to create database per role.
If you want to implement that with one database where only a set of users can read some and not all documents, you must use a proxy server (Application layer) which wouldn’t be very “Couchy”. The simplest way would be to create a “_list” function, add all logic to the “_list” function and call the list using the proxy. Of course, you will need to add extra fields for each document specifying who can read/write this document. This would add extra complexity and slow down everything as list function would deal with documents one by one. For the replication part you will need to use filtered replication with some query_params.

Reply



Mirza Abazovic on March 23, 2017 at 8:11 am
Thank You for answering

Reply

emiliano on December 19, 2016 at 7:41 pm
Sorry for the question, but i don't know how to install the plugin couchperuser. Yes, i see the steps described in the github page, but still i don't know how to do it. Is it possible that you can tell me how to do it? Thanks!
Reply

Fady S. Ghatas on December 22, 2016 at 8:47 pm
Hi Emiliano, I would need extra information to be able to help. What's your distribution? Does any error message show up?
Reply

Tim on January 28, 2017 at 5:41 pm
Great Post
Reply

NoSQLer on March 4, 2017 at 10:15 pm
Amazing post. Do you still use Couch?
Reply

Fady S. Ghatas on April 25, 2017 at 5:16 pm
For some projects I still use CouchDB, but I've replaced it with SQL for some projects as well, check this : http://www.titrias.com/abandoning-couchdb-nosql-favor-sql/
Reply

Bastian Torres on July 11, 2017 at 2:22 am
Hey Fady, great article!
I tried to implement this "database-per-user" architechture with no success. Maybe you can help me unlock my path. This is my scenario: I've started working with a new fresh installation, so I've turned off the "Admin Party" feature, created an admin user and enabled CORS.
I'm using Couch V2, so the database-per-user it's just a flag that you can enable from the Configuration tab within the Fauxton admin panel. I'm on Mac.
The first thing I've noticed is that I had to manually create the _users database. Without this, the user creation proccess failed because there was no _users database to add in. Maybe you can add that

detail to your post.
Secondly, and most important thing, when I create a new user (I'm using Postman to hit the Couch API), I'm seeing that the user is created, but I dont see the user database. Why is this? I have to manually create the user's database too? I thought that this was done automatically when a new user is created…
Thank you for your time!
Reply

Fady S. Ghatas on July 11, 2017 at 2:50 pm
Hello Bastian,
Database-per-user is NOT working in CouchDB V2. Ex : https://gist.github.com/nolanlawson/9676093
So you may downgrade to 1.6 and use the database-per-user daemon, or manage it manually (Create the user, then create the database, then manage permissions of the new database), this is not a good solution for multiple reasons : portability, maintenance and most importantly this process won't be atomic, so you will need to handle each step exceptions and fallback if needed.
Hope this helps, please do not hesitate to contact me if you have any further questions.
Reply

Bastian Torres on July 27, 2017 at 2:31 pm
Thank you Fady! Finally I've decided not to downgrade to Couch 1.6, because the Fauxton admin panel already installed in Couch v2 is far more consistent and user-friendly.
The implementation of the database-per-user using just the native Couch roles system was far more easy than I believed. The only cuestionable thing is that I had to implement a server-side endpoint to manage this configuration when a new user/db is created, but is fine for my requirements.
Thanks for your help!
Reply

Fady S. Ghatas on August 1, 2017 at 9:32 am
Correct, the process is fairly easy, just make sure to handle the atomicity at least so as not to be left with a user without a database or a database not linked to a user. This will require manual fixing if not

handled correctly.
Reply

Serg on November 29, 2018 at 6:13 pm
Thank you!
Reply

Fady S. Ghatas on December 1, 2018 at 1:30 pm
You're very welcome, thanks for stopping by.
Reply

Ben on November 21, 2019 at 12:48 am
hi, nice post, but what about if i have have multiple “tables” per user too. do i have to create multiple databases per user (for each table, like tasks, games,) or should i make them a document and put the values in an array. (thus having online one database per user) ?
Reply

Fady S. Ghatas on December 12, 2019 at 8:28 am
A single database per user is the better approach. And use map/reduce and a type field to accomplish what you have in mind.
Reply

Hasan on January 17, 2020 at 1:24 pm
nice post. I have come from MongoDB and Oracle Sql background. I will be glad if you answer:
If CouchDB has a db which has 5Million docs. Total user 10Million need some of those docs. So to make replication work with PouchDb, I need to create 10 Million db in couchDB(per db per user)?
Reply

Fady S. Ghatas on January 19, 2020 at 5:07 pm
Exactly, 10,000,000 DB.
This might be intimidating at first but it should work just fine. One thing to take care of though, you will need to do some kind of manual partitioning on files (the users's DB files). You can split names using slashes which will cause the OS to create subdirectory on a slash, which will help the filesystem reaching the appropriate user file faster. Similar to when you do indexing in Oracle Sql to make searching for rows faster and Oracle

automatically creates B-Tree.

Reply