

Forum:[Hack Forums](#)

Thread: [NoSQL: Pure PHP DataBase System Class V0-BETA](#)

Ego Inc - January 21, 2015, 1:41 pm

V5 is out: <https://hackforums.net/showthread.php?tid=4662288>

Angel Beats?? - January 21, 2015, 1:46 pm

This is a nice project man, i'd like to see some benchmarks vs sql or other dbs.

I was thinking of doing this a while back but never started :(

Ego Inc - January 21, 2015, 1:48 pm

(Jan 21, 2015 02:46 PM)Ang?! Beats Wrote: [\[CITE\]](#) This is a nice project man, i'd like to see some benchmarks vs sql or other dbs.

I was thinking of doing this a while back but never started :(

Working on that, It will come out along with a more in depth documentation here soon. And yeah it was a lot of fun.

TRH4CK - January 21, 2015, 1:50 pm

That looks awesome.

Vouch for Tux. he is a mysql & php pro!

Angel Beats?? - January 21, 2015, 1:50 pm

(Jan 21, 2015 02:48 PM)Tux. Wrote: [\[CITE\]](#) Working on that, It will come out along with a more in depth documentation here soon. And yeah it was a lot of fun.

If you could make it take commands like mysql that would be amazing.

Ego Inc - January 21, 2015, 1:53 pm

(Jan 21, 2015 02:50 PM)Ang?! Beats Wrote: [\[CITE\]](#) If you could make it take commands like mysql that would be amazing.

Meh I thought about it, structuring it with functions instead was much easier tho as that would be like writing a new language/interpreter in many respects. This way also injections aren't a problem.

hellomen - January 21, 2015, 9:00 pm

Hooray to get back to earlier dates when the first databases were made xD

It's nice to see you doing this tho'.

Haxalot - January 22, 2015, 5:11 am

It's a terrible name to have for a simple flat-file database to be honest.

General comments:I can't see this scaling well at all since you are creating a new file for every single row, which means you're having to dip into every single file just to check each row's contents. Even the primary key (the id) of the row cannot be optimised because of this design (effectively making the primary key a unique thing only, rather than an optimised way to perform lookups too).

Your naming conventions are cryptic and inconsistent. For example, consider mktb, which looks like a string of pseudo-random string of characters to remember from a developer's perspective. And then the get_cl_id method, which uses snake case with the verb in full form separated from the rest. And then rmcl_by_id, which uses the verb in shorthand form combined with another string of

characters ("cl").

This isn't easy to remember because of inconsistency and unintuitive naming conventions. I'd recommend you adhere to the PSR-1 naming conventions of method names by using studyCaps. So your method names above would change like so:

```
mktb => createTable  
get_cl_id => getColumnId  
rmcl_by_id => deleteColumnById
```

The few extra characters used makes your code self-documenting, which any programmer will be a lot more thankful for. Also, I've used the names corresponding to the CRUD operations since they're more ubiquitously used and therefore feel more intuitive than 'make', 'read', 'update', and 'remove'.

Try to adhere to PSR-* standards in general. Your code will be more readable for it.

There's a number of other improvements you could do too, but I don't have time to talk about them for now.

Ego Inc - January 22, 2015, 12:34 pm

(Jan 22, 2015 06:11 AM)Haxalot Wrote: [\[CITE\]](#) It's a terrible name to have for a simple flat-file database to be honest.

General comments:I can't see this scaling well at all since you are creating a new file for every single row, which means you're having to dip into every single file just to check each row's contents. Even the primary key (the id) of the row cannot be optimised because of this design (effectively making the primary key a unique thing only, rather than an optimised way to perform lookups too).

Your naming conventions are cryptic and inconsistent. For example, consider mktb, which looks like a string of pseudo-random string of characters to remember from a developer's perspective. And then the get_cl_id method, which uses snake case with the verb in full form separated from the rest. And then rmcl_by_id, which uses the verb in shorthand form combined with another string of characters ("cl").

This isn't easy to remember because of inconsistency and unintuitive naming conventions. I'd recommend you adhere to the PSR-1 naming conventions of method names by using studyCaps. So your method names above would change like so:

```
mktb => createTable  
get_cl_id => getColumnId  
rmcl_by_id => deleteColumnById
```

The few extra characters used makes your code self-documenting, which any programmer will be a lot more thankful for. Also, I've used the names corresponding to the CRUD operations since they're more ubiquitously used and therefore feel more intuitive than 'make', 'read', 'update', and 'remove'.

Try to adhere to PSR-* standards in general. Your code will be more readable for it.

There's a number of other improvements you could do too, but I don't have time to talk about them for now.

Harsh as that was, thank you. I'm working on the next version of this right now and implementing

your suggestions as I can.

Graphic. - January 22, 2015, 12:54 pm

Vouch for Tux.

Thanks for sharing.

hellomen - January 22, 2015, 2:53 pm

How could NoSQL be a terrible name if it's the original name for such?

Google NoSQL I believe it was invented in 2000

MaskingMyUsername - January 22, 2015, 3:25 pm

Well seeing as you begun this as a fun project, the naming conventions aren't a massive issue.

You've done good here, it's a really nice system, and it's great to see stuff like this being shared here. Can't wait to see future versions!

itzDecon - January 23, 2015, 9:20 pm

I've had ideas about this in the past, but have never attempted to actually complete such a project. However, I have a suggestion.

From looking over the file, it appears each DB gets it's own directory, then each table get it's own directory inside of that, then each row of data get's it's own file, plus the additional .htaccess, .owner, etc files. An idea to clean all this up, would be to make a directory for the whole system, then each database gets it's own (XML) file, then continue down from there so that the file would look something like:

```
Code <?xml version="1.0" encoding="utf-8"?>
<database>
  <table name="members">
    <row>
      <id>1</id>
      <username>Username</username>
      <password>qwerty</password> [b]<!--[/b] Please don't actually store plain text passwords...
Or use shitty passwords... -->
    </row>
    <row>
      <id>2</id>
      <username>Admin</username>
      <password>password1</password> [b]<!--[/b] Please don't actually store plain text
passwords... Or use shitty passwords... -->
    </row>
  </table>
  <table name="settings">
    <row>
      <key>base_url</key>
      <value>http://localhost</value>
      <default>http://www.google.com</default>
    </row>
  </table>
</database>
```

Ego Inc - January 24, 2015, 4:09 pm

(Jan 23, 2015 10:20 PM)itzDecon Wrote: [\[CITE\]](#) I've had ideas about this in the past, but have never attempted to actually complete such a project. However, I have a suggestion.

From looking over the file, it appears each DB gets it's own directory, then each table get it's own directory inside of that, then each row of data get's it's own file, plus the additional .htaccess, .owner, etc files. An idea to clean all this up, would be to make a directory for the whole system, then each database gets it's own (XML) file, then continue down from there so that the file would look something like:

```
Code <?xml version="1.0" encoding="utf-8"?>
<database>
  <table name="members">
    <row>
      <id>1</id>
      <username>Username</username>
      <password>qwerty</password> [b]<!--[b] Please don't actually store plain text passwords...
Or use shitty passwords... -->
    </row>
    <row>
      <id>2</id>
      <username>Admin</username>
      <password>password1</password> [b]<!--[b] Please don't actually store plain text
passwords... Or use shitty passwords... -->
    </row>
  </table>
  <table name="settings">
    <row>
      <key>base_url</key>
      <value>http://localhost</value>
      <default>http://www.google.com</default>
    </row>
  </table>
</database>
```

Thanks I've actually already began updating it and it's running much faster and is much cleaning now. Expect v1 to be out in the next few days with benchmarking against MySQL.

Haxalot - January 26, 2015, 4:15 am

(Jan 22, 2015 03:53 PM)hellomen Wrote: [\[CITE\]](#) How could NoSQL be a terrible name if it's tge original name for such?

Google NoSQL I believe it was invented in 2000

The term NoSQL was coined back in '08, I think. It's not a single technology either, but rather a term to describe a group of database - so there's no 'invention' date. NoSQL doesn't really have a definition associated with it, and with no authoritative body to give one, it will likely never be properly defined. Instead, there's common characteristics usually shared by the group of data stores that come under the NoSQL umbrella:

Open SourceNoSQL data stores have deep roots in the open source community. There are of course exceptions to this, though mostly you'll see NoSQL data stores as open source projects.

Cluster-OrientedNoSQL data models mostly follow the notion of being aggregate-oriented, allowing the data they're storing to be distributed over a cluster of machines - thereby providing horizontal

scalability. The exception category of NoSQL data stores is Graph, which are more similar to relational databases but follow a different data model. This alternate model actually makes them better at handling complex relationships within the data persisted than relational databases.

SchemalessNoSQL data models don't have a fixed schema in place. This is the complete opposite to relational databases that have a very rigid and fixed schema in place. There's both positive and negative effects of having a schemaless design, and they will be mentioned later on.

Created in the 21st CenturyNoSQL data stores first arose from the need to process and store larger quantities of data. This happened in the 21st century with the dot-com boom, and web applications saw a dramatic increase in traffic - both from web users and, later on, bots. One effect of this was that it became more valuable to capture data about web users for analysis - this soon made it clear that there were serious limitations for data storage and processing for the largest companies like Google and Amazon.

Not using the Relational Model, and therefore not using SQLNoSQL data stores can be broadly (and crudely) grouped into four categories of data models: Document, Key-Value, Column-Family, and Graph. None of these (at the time of writing this) use the SQL language - namely because SQL is designed to work with the relational model. The language is therefore deeply rooted in relational theory where the mathematical theory of sets drive the principles of SQL's constructs. Since NoSQL data stores reject the relational model (and therefore have no concept of relations), this language isn't used to communicate (as queries) with NoSQL data stores. They instead have their own languages, as we shall see later on.

So I stand by my point - NoSQL isn't a good name for a flat-file database.

clickosface - January 28, 2015, 11:19 am

tLooks very good. Well done.
tanks ty

Ego Inc - January 28, 2015, 12:16 pm

(Jan 28, 2015 12:19 PM)clickosface Wrote: [\[CITE\]](#) tLooks very good. Well done.
tanks ty
Naw it's shit. Much better version coming soon tho.

3mmi - January 29, 2015, 10:27 am

So far so good mate. Keep up the good work, this is interesting.

ShadowCloud - January 29, 2015, 10:38 am

Firstly, this is a really valiant attempt at something that's actually extremely complex. Whether this works or not, I applaud you for actually putting your head down and getting into it.

Haxalot has provided some really good feedback and I absolutely read it as constructive feedback, I can only hope you do as well ;)

That being said, here's my concerns for using this:

The one thing you absolutely have to find a way around is a locking mechanism for flat files. You are bound to have concurrency issues if you don't address this directly. How will you handle

deadlock conditions and/or dirty reads?

Caching mechanisms will be vital so you can read frequently accessed data from a cache, preferably keeping small pieces in memory?

Have you considered some form of indexing to speed up retrieving specific data?

I'd like to have a way to fetch a list of rows rather than only being able to fetch a row with a specific ID. In most cases I won't know the ID either... I'm getting a pastebin removed message from the OP, so I can't see if you have a function for this?

How are you handling the load of existing items? If I have 2GB of data, is this being initialized when I create an instance of the DB?

Again, overall you have a long way to go, but this is way better than what I expected when I saw the thread title. Good job ;)

Support Operator - *January 29, 2015, 1:01 pm*

You should store all the data in JSON rather than a new file per row etc. Also, you shouldn't read the file each time a lookup is done... you should store it in a variable and then as the `__destroy()` method; save it.

Other than that, good idea... not the best execution.