

Non-relational databases

In this article I will show an overview and basic use of some alternative styles of databases, more specifically the non-relational databases SimpleDB and CouchDB. These can have distinct advantages over more typical databases in particular circumstances. I will give some background to each database, a guide to get you setup and some basic code samples to get you started.

What are typical databases? Most databases a PHP developer would normally find themselves working with are databases such as MySQL, PostgreSQL or SQLite. These are all commonly available, MySQL being the commonest used with PHP. There are of course other relational database options, for example:

- SQLite - a simple file based database with a lot of features including zero configuration and transactions. It is enabled by default as of PHP 5 - <http://www.sqlite.org/>
- PostgreSQL - touts itself as the world's most advanced open source database - <http://www.postgresql.org/>

Commercial relational databases;

- Oracle
- IBM DB2
- Microsoft SQL Server

Normally when we develop web applications there is usually a need to store some data that needs to last longer than just a session or cookie would last. Most PHP developers will have already used MySQL for this job - as it is one of the most common relational databases around.

We use databases in our applications to store structured data about these things - what products a store has, blog posts / comments. Relational databases allow grouping of a dataset around key attributes - that is they allow you to define how different rows relate to each other. For instance should you have an database which stores name and address details for your clients you may have a schema which looks something like this;

```
Person
-----
id
firstname
surname
email
password
dob
```

```
Address
-----
id
person_id
type
address_1
address_2
address_3
city
```

state
zip

In this example we are storing related data using a key - a unique attribute against a person. This allows us to reference that key against each person's address thus relating them together - this is a fundamental concept of a relational database.

So, now that we have covered the basics of how a more traditional database stores data - what are the alternatives do we have and why would we consider them?

Document based databases

Document databases differ from relational databases as they are designed to store related attributes together in a single record - much like a sheet of paper. They are designed to store, retrieve and report on semi-structured data - there is no enforced schema. This means that documents take up only the actual space they use as well as avoiding time consuming schema updates for changes.

In most web applications life time the database schema will change - additional unthought of requirements will appear and changes will have to be made. Usually this involves updating every record by altering the schema. With no schema to be enforced in the first place different documents may have different attribute sets with no issues.

CouchDB and SimpleDB also support distributed operation. In SimpleDB this is not an option - your data is distributed amongst many servers at Amazon to keep it safe and ensure speedy access.

CouchDB

CouchDB is a document database written using the functional programming language Erlang. Documents are stored in JSON (JavaScript Object Notation) and retrieved via HTTP, which may at first seem alien compared to what you are familiar with.

However once you stop and think about it this makes its use quite simple as most modern languages have built in functions for dealing with HTTP requests as well as JSON built in - this makes it easy to access data from almost language. As the database is accessed via HTTP there is the possibility of improving performance by using the same techniques as a you would for a web server - for instance by using a reverse proxy such as Varnish or Squid.

CouchDB aims to be distributed and fault tolerant - these features are much like MySQL replication, allowing a complete copy of a database on another machine. Machines can be manually synchronized or recently on trigger basis. CouchDB supports bi-directional replication of databases - changes are synchronized both ways. This can allow for interesting uses such as offline replicas, with conflicts being automatically resolved where possible.

CouchDB has another notable feature - views. CouchDB views are much like MySQL views - they take data and report on it with the structure not necessarily being the same as the source. CouchDB views are written using Javascript map/reduce functions. By default they are updated (or built) on access rather than when data is inserted. Unfortunately Views are beyond the scope of this article - there is simply not the space to go in to enough detail to be useful. However there are several useful links in the further reading section.

Getting ready : Getting the prerequisites, source and installing

First of all we need some prerequisites so we can build CouchDB from source. This article is being written using a Debian based machine - so the following commands may need to be substituted if you are using something non-apt based - there are instructions in the README file in the trunk which may help.

From your shell run the following commands - these will install the subversion tools, Erlang language and some build related tools;

```
apt-get update
apt-get install subversion
apt-get install automake
apt-get install autoconf
apt-get install libtool
apt-get install help2man
apt-get install build-essential
apt-get install erlang
apt-get install libicu-dev
apt-get install libmozjs-dev
apt-get install libcurl4-openssl-dev
```

Now to we need to checkout the latest trunk version of CouchDB from the Apache subversion repository;

```
cd /usr/src
svn co http://svn.apache.org/repos/asf/couchdb/trunk couchdb
```

You should see some output - which if successful will end in something like "Checked out revision 779085".

We are now ready to start building the source files - first we need to bootstrap the code;

```
cd couchdb
./bootstrap
```

If you get any errors at this stage they are usually caused by missing dependencies and can usually be resolved quite easily. Tip: If you cannot install the missing package using just "apt-get install <package>", try "apt-cache search <package>" to try and find the exact name of the package you need to install.

Next we run configure:

```
./configure
```

Tip: If you want to change the default install location (which is /usr/local/) you can do so by changing the configure options. Normally you should not need to alter the defaults. To list the options type:

```
./configure --help
```

Next is actually building the code - if your computer has more than one core / cpu you can speed this up by changing 4 to the number of cores you have. I've left it on four as I have a quad core box.

```
make -j 4
```

And finally installation:

```
make install
```

Now that everything is built and installed, we are ready to start up Couchdb for the first time. Currently CouchDB will bind to localhost only - so if like me you are testing using VMWare you will need to alter the interface that CouchDB binds to. This is done by changing the bind_address setting in the file /usr/local/etc/couchdb/local.ini - you will need to know which IP address you want to bind to.

Starting CouchDB for the first time by typing;

```
couchdb
```

You should receive a response like this:

```
Apache CouchDB 0.10.0a779085 (LogLevel=info) is starting.  
Apache CouchDB has started. Time to relax.  
[info] [<0.1.0>] Apache CouchDB has started.
```

The Futon web interface

CouchDB ships with a web interface which is available by default at http://<couchdb ip>:5984/_utils/

Futon allows you to create databases, manage documents and do various administration tasks including running tests and setting up of replication. For now though we are just going to use it to do some simple examples with PHP.

Firstly, create a new database - we'll be using this later. We'll call it 'phparch' as we will build our example to store records about PHP Architect issues - to do this just click "Create Database...".

Next, create our first document - to do this click on your newly created database and then click the "Create Document" link. When asked for the document id enter "test".

We should now see the document detail page - by default every document has the two attributes `_id` and `_rev`. `_id` is used to uniquely identify each document - `_rev` is an internally used attribute for pertaining to the revision of the document - its used internally by CouchDB.

Getting started : The PHP part

It's quite simple to write some basic code to interface with CouchDB due to its nature - requests are via HTTP. This means very simple tasks can be done just by using the fopen

built in url wrappers. For instance we can retrieve the document just added with the following code;

```
<?php
$doc = file_get_contents('http://<couchdb ip>:5984/phparch/test');
echo $doc;
?>
```

This will output something similar to this;

```
{"_id":"test","_rev":"1-231417934"}
```

Like this however it is probably not of much use - this is a JSON encoded string. Luckily since PHP 5.2 there are several built in functions to deal with JSON available to us. To get the document in to a more usable format we need to decode it;

```
<?php
$doc = file_get_contents('http://<couchdb ip>:5984/phparch/test');
$doc = json_decode($doc);
print_r($doc);
?>
```

This will now allow us to do things such as `$doc->_id`, which is much more useful.

Its quite possible for us to write our own wrapper to communicate with CouchDB but this is beyond the scope of this article. Fortunately there is a good library already written in the form of PHPillow - an object orientated CouchDB wrapper written by Kore Nordmann (http://kore-nordmann.de/about_me.html).

Amazon SimpleDB

Amazon SimpleDB is an offering by Amazon Web Services - it is a commercial service which aims to be a cheap way of getting reliable, scalable and on-demand database infrastructure to the masses. It provides a simple equivalent to a clustered relational database without having to shell large amounts of money on initial setups, database and server administrators, hardware for servers and infrastructure as well time spent maintaining and troubleshooting an in-house solution.

It does this by providing a service which much like CouchDB requires no fixed schema and has a simple to interface with API for accessing your data. SimpleDB also automatically manages the replication and indexing of your data so you don't have to.

As with all services from Amazon Web Services, you only pay for what you use - there is no monthly minimum or setup charges to worry about. What exactly are you charged for? Charges for SimpleDB are made for three things - storage of your data, transfer of data between Amazon and your server and lastly for Machine hours. Machine hours can be thought of as processing time for your queries - the longer a query takes the more expensive it will be.

Price wise, at least for this introduction the service should be free to use. Starting from December 2008 Amazon have introduced a free pricing tier which includes the first 25 Machine hours and the first gigabyte of storage and transfer per month. They estimate that this roughly equates to 2 million average requests.

To learn more about the pricing please read this page : <http://aws.amazon.com/simpledb/#pricing>

If you are interested in trying SimpleDB you will need to have your own AWS account and also be signed up for SimpleDB - there is no setup cost involved however you may need a credit card.

<http://aws.amazon.com/simpledb/>

Getting ready : Downloading the libraries & prerequisites

Once you have successfully signed up for your AWS account and added SimpleDB to your account you will need to download a client library and have your AWS access identifiers handy. These are two short text strings which you can find in the "Access Identifiers" section in your AWS account.

The Access Identifiers are used to sign any requests you make to any AWS service - this ensures that you have a valid account and that no one else can make charitable requests on your behalf. It is important you keep these secret.

To generate these hashes we need to install the Pear HMAC library - assuming you are on a Debian / Ubuntu based machine installing PEAR is very easy;

```
sudo apt-get install php-pear
```

And installing the HMAC library is also simple;

```
sudo pear install Crypt_HMAC
```

You will also need the client library for SimpleDB - there are a number of choices available at the moment. We will be using one of the simpler libraries. You can download it from the following link;

http://sourceforge.net/project/showfiles.php?group_id=253828

Getting started : The PHP part

After you've downloaded and uncompressed the SimpleDB library you're pretty much done - the only configuration that you need to do is when you instantiate a copy of the library. For this you will need to replace your Access Identifiers in the following script;

```
<?php
require_once 'sdb.php';

$sdb = new SimpleDB('<your access key>', '<your secret key>');

$domains = $sdb->listDomains();
```

```
print_r($domains);
?>
```

If you run this script having never used SimpleDB before you should not get much output - this is expected as we've not created any Domains yet. Domains can be roughly thought of as Tables / Databases - the main principal is that they can be used to break up big things such as applications.

So, having established that we do not have a Domain yet, we need to create one - domains take alpha numerical strings as identifiers - we are going to create one called 'phparch', which the following code will do;

```
<?php
require_once 'sdb.php';

$sdb = new SimpleDB('<your access key>', '<your secret key>');

$sdb->createDomain('phparch');

$domains = $sdb->listDomains();

print_r($domains);
?>
```

So now we have created our new domain we need to know how to insert entries - thankfully this is also quite simple. We are going to add information about the various magazine issues we have - this is done as follows;

```
<?
require_once 'sdb.php';

$sdb = new SimpleDB('<your access key>', '<your secret key>');

$sdb->putAttributes('phparch', 'may', array('title'=>array('value'=>'May
2009'), 'have'=>array('value'=>false)));
$sdb->putAttributes('phparch', 'june', array('title'=>array('value'=>'June
2009'), 'have'=>array('value'=>true)));
$sdb->putAttributes('phparch', 'july', array('title'=>array('value'=>'July 2009'),
'have'=>array('value'=>true)));
?>
```

Putting attributes is the SQL equivalent of INSERT INTO / UPDATE - either inserting or updating an existing entry in a Domain.

Amazon now support a SQL like interface which most developers will find has a shallower learning curve especially coming from a MySQL background. The next example runs such a query and only returns issues we marked as having:

```
<?
require_once 'sdb.php';

$sdb = new SimpleDB('<your access key>', '<your secret key>');
```

```
$r = $sdb->select('phparch', 'SELECT * FROM phparch WHERE have = "1"');  
  
print_r($r);  
?>
```

This short introduction to SimpleDB has covered some of the basics, if you are interested in learning more I would suggest reading through the SimpleDB class and checking out the Developer resources (see Further reading).

Conclusion

Although not so commonly used, non relational databases have their place in our lives as developers. Like all databases they are good in some areas and not in others - as with most things choosing the right tool for the job is essential. Hopefully the quick over view has given you a look in to what you can do with databases designed from the ground up for a specific purpose.

Further reading

Should you wish to read more about SimpleDB or CouchDB these links should give you a good insight;

- CouchDB: The Definitive Guide - an as yet unpublished book due in September 2009
- CouchDB's wiki provides lots of reference information on basic to advanced topics - <http://wiki.apache.org/couchdb/>
- View snippets - http://wiki.apache.org/couchdb/View_Snippets
- More on views and "JOINS" in CouchDB - <http://www.cmlenz.net/archives/2007/10/couchdb-joins>
- If your looking to connect and chat to some real life users #couchdb on the freenode IRC network is recommended

For SimpleDB;

- Amazon's SimpleDB product page - <http://aws.amazon.com/simpledb/>
- Developer resources - <http://docs.amazonwebservices.com/AmazonSimpleDB/2007-11-07/DeveloperGuide/>
- What you need to know about SimpleDB - <http://www.satine.org/archives/2007/12/13/amazon-simpledb/>