Not just an abstraction layer

The Official Cross Database Engine Guide

INTRODUCTION

Cross Database Engine started as a need arose in our company to work with multiple databases from a singular source base. It was frustrating to change function calls when the database backend changed for our systems. In cases where the database didn't change but new libraries appeared it was also basically a "rewrite" of some systems.

If you're reading this I'd like to encourage you to dive in on the examples which follow and you'll see that CDE lives up to its slogan "Not just an abstraction layer". If you are a serious web developer in PHP you'll find that CDE will help you get the edge in your coding.

If you have been using CDE and want me to send you a message when we're releasing new changes then please feel free to send me an email at andre@spiceware.co.za with CDE Update in the subject and a brief description about who you are and how you want to use CDE. Spiceware the company helped started in 2003 works with CDE every day as our base database platform for database connectivity.

Thank you for using CDE and enjoy the examples below of how to get started.

CONTENTS

Not just an abstraction layer

The Official Cross Database Engine Guide

DO THIS FIRST

Extract cdesimple.php from the zip archive and place it in your "includes" folder on the server or simply in the root of the website or application you are building. Following that you simply require it in your connection file or index file.

```
require_once "cdesimple.php";
```

EXAMPLE 1 – CONNECTING TO A DATABASE

The best way to learn a new tool is to see how it works in practice. In all the examples I will be connecting to a SQLite database but I will make mention or show some examples of connecting to other databases so you can see how it would work. MSSQL and ORACLE connection strings are pretty tricky and normally it is the PHP configuration that needs to be done properly in order to get those working, especially if you are using IIS as a web server. Please feel free to contact us if you are struggling, we have done this all before!

```
//Example of a connection

$database = "test.db";
$username = "";
$password = "";

$CDE = new CDESimple ($database, $username, $password, "sqlite3", $debug=false,"dd/mm/YYYY");
```

The above connection will create a database called test.db relative to the script file that called CDESimple. If you want to make it more specific, pass the full path to the database parameter.

In this example we connect to a Firebird or Interbase database, we should make sure the database exists before we create the connection. I am just using the defaults here, please never use the default username and password on a production server. Notice how the address of the server is prefixed with a trailing ":" on the database variable (hostname:path).

```
//Example of a Firebird connection

$database = "127.0.0.1:/home/firebird/TEST.FDB"; //make sure the database exists first
$username = "SYSDBA";
$password = "masterkey";

$CDE = new CDESimple ($database, $username, $password, "firebird", $debug=false,"dd/mm/YYYY");
```

In the following example we connect with a MySQL or MariaDB database, in the backend CDE will determine which PHP functionality to use for the system. In the database variable we specify the hostname and database name and can if we wish specify a port if we wish (hostname:path:port);

```
//Example of a Firebird connection

$database = "127.0.0.1:MYSQLTEST"; //make sure the database exists first
$username = "root";
$password = "masterkey";

$CDE = new CDESimple ($database, $username, $password, "mysql", $debug=false,"dd/mm/YYYY");
```

CODING RECOMMENDATIONS

Place the require_once for cdesimple.php and declare the $CDE object as global in a "connection.php" file which can be required in all your scripts that need it when you need to communicate with the database.

EXAMPLE 2 – RUNNING SOME SQL

The hardest part is over if you are able to connect to the database as per Example 1. The following example is to use the exec method of CDE to run SQL statements. Generally you will want to create some tables in your database once you have created it. I refer specifically to SQLite now but the principal is the same for most databases.

```
//Example of creating a table and adding some data

$usertable = "create table if not exists tbluser (
                      userid integer default 0 not null,
                      name varchar (200) default '',
                      email varchar (200) default '',
                      passwd varchar (255) default '".crypt("changeme")."',
                      status integer default 0 not null,
                      image blob default null,
                      datecrt date default null,
                      primary key (userid)
                    )";

$CDE->exec ($usertable); //Execute the statement

$CDE->exec ("insert into tbluser (userid, name, email, passwd, status, datecrt) values (0,
'Admin', 'admin', '".crypt('changeme')."', 1, 'now')");
```

Notice the use of 'now' to indicate the default date, I come primarily from a Firebird background and CDE will understand that I wanted to use Firebird SQL even though I am talking to a SQLite database. CDE will try to understand for example if you said "select top 100 from table" on a Firebird database it will translate your instruction to "select first 100 from table" without you being aware that you even made a mistake.

The exec method can take parameters also which make it really easy to "upload" files into your database.

```
//Example of using a parameter, a parameter is a ?

$CDE->exec ("update tblimage set imagedata = ? where imageid = ?",
file_get_contents("logo.png"), 1);
```

Some databases require you to commit your statements after execution if you want to make sure the data is saved or in case you are unsure if your script may not work until the end or time out.

```
$CDE->commit(); //make the transaction stick (not applicable to SQLite or MySQL)
```

CODING RECOMMENDATIONS

We all know that dates can be a headache and that users will insist on putting funny characters in their inputs. CDE has the solution for these.

Dates:
Decide which date format you want to use on offset example “YYYY-mm-dd” then use the date_to_db method to put your date (2012-12-01) into the exec statement.

```
//Example of date_to_db
$CDE->exec ("update tbluser set datecrt = '".$CDE->date_to_db("2012-12-01 22:00")."' where
userid = 1");
```

Funny characters:
All those things that break your SQL can be avoided.

```
//Example of string
$CDE->exec ("update tbluser set name = '".$CDE->escape_string($name)."' where userid = 1");
```

We hope these methods will help you not have to go through crazy debugging to work out what is going wrong when inserting data.

EXAMPLE 3 – RETRIEVING DATA – READ CAREFULLY TO AVOID DISSAPOINTMENT!!!

The data is now in the database, you can run all your insert, update and delete statements using the exec method and now you need to retrieve the data. You must understand the reasoning behind how data is retrieved in CDE to use it properly. I discovered with training developers and using databases across different code sets that the data and field information would always come back in an unexpected manner and to this end I decided the following:

- All record results would be returned in either an object, array or associated array format
- Field names would be UPPERCASE so the developer did not need to make mistakes
- Date fields would be setup in the format specified as per the connection so the developer knew what format he was getting his dates in.
- Blob fields would automatically be included in the result (can be turned off)

If you comprehend the above statements you will immediately understand why you should be using CDE as a solution for your website or application.

```
//Example of getting a result as an associate array (good for where you make spaces in your
field names ☺

$users = $CDE->get_row ("select * from tbluser", CDE_ASSOC); //can be CDE_OBJECT or CDE_ARRAY

if (!file_exists ('output')) {
  mkdir('output');
}

foreach ($users as $uid => $user){  //we use a foreach statement, how easy can it be?
  echo "Name : {$user["NAME"]} \n";
  echo "Surname : {$user["SURNAME"]} \n";
  echo "Email : {$user["EMAIL"]} \n";
  file_put_contents ("output/image{$user["NAME"]}.png", $user["IMAGE"]);
  echo "<img src=\"output/image{$user["NAME"]}.png\" />\n";
}
```

The above example illustrates working with an image field in your table where image data has been uploaded. If you need to see more on this then have a look at the cdetestclass.php for more ideas.

EXAMPLE 4 – FIELD INFORMATION ON A RESULT

You may have noticed like I did that every database implementation gives different methods in retrieving field information from the query you have just run. You may also have noticed that you were unable to retrieve field information for say SQLite3??? Well now that is all possible with CDE and you don't have to worry when you change your database backend. The field information is generated automatically when you use the get_row method or get_value method and can be accessed like this:

```
//Example of getting the field information

$fieldinfo = $CDE->fieldinfo;

echo "<pre>";

print_r ( $fieldinfo );

echo "</pre>";
```

And you should see on the screen something like this:

```
Array
(
    [0] => Array
        (
            [0] => DOCTORID
            [name] => DOCTORID
            [1] => DOCTORID
            [alias] => DOCTORID
            [2] => TBLDOCTOR
            [relation] => TBLDOCTOR
            [3] => 4
            [length] => 4
            [4] => INTEGER
            [type] => INTEGER
            [5] => right
            [align] => right
            [6] => 120
            [htmllength] => 120
        )

)
```

The above is just a single field from the result, as you can see there is some important information that is returned notwithstanding the actual field name. The alias for example is returned as well as the field name if you had a select statement which used an as suffix when you wrote your select statement. This is also necessary if you write sub select statements like we do and want to reference them properly in the result set. Also the field type is returned and the suggested HTML width and align styling to use based on the type.

CONCLUSION

I will be updating this document with more of the advanced functionality of CDE but this should suffice in getting you started and get you well on the way to making robust cross database websites and applications.