MantisBT Database Layer

The MantisBT Database layer consists of two components:

- MantisDatabase Class
- MantisDatabaseDict Class.

Database Engines

The MantisBT Database layer utilises PHP's PDO layer and supports the following Database engines

Name	PDO Driver	Known Supported Versions
Oracle	pdo_oci	Oracle Database XE 11.2
MSSQL	pdo_sqlsrv	Microsoft SQL Server 2012
PGSQL	pdo_pgsql	Postgresql 9.3.4
Mysql	pdo_mysql	Mysql 5.6.15

> **Note**
>
> There are PDO Drivers for other database engines. In theory, to add a new database engine would require a new MantisDatabase and MantisDatabaseDict class for the required engine. However, we plan to only support the 4 engines listed above to ensure a quality experience for end users.

MantisDatabase Class

The following code example demonstrates how to make use of the MantisDatabase class:

```
$g_db = MantisDatabase::get_driver_instance($g_db_type);
$t_result = $g_db->connect( null, $g_hostname, $g_username, $g_password, null, null );
if( !$t_result ) { ... }
$g_db->SetPrefixes( $g_prefix, $g_suffix );
```

This performs 3 basic functions:

- Get A MantisDatabase Driver instance for one of the supported Database Connections
- Attempt to connect to the database
- Set Prefixes for Tables

From this point, it's possible to use the execute function for example to run a database query.

```
$g_db->execute($sql, array $params=null)
```

A complete list of functions provided by the Database Class are outlined below:

- SetPrefixes
- Get_server_info
- driver_installed
- get_driver_instance
- get_dbtype
- diagnose
- connect
- dispose
- get_last_error
- get_tables
- get_indexes

- get_columns
- get_insert_id
- set_debug
- execute
- SelectLimit
- table_exists
- index_exists

->SetPrefixes($p_prefix, $p_suffix)

The SetPrefixes function takes 2 parameters:

a) A prefix to use for all database tables b) A suffix to use for all database tables

In short, this means that for

->get_server_info()

This returns an array of server information of 3 components:

- version
- description
- information

The version field is expected to be an integer format that can be passed to version_compare

```
$t_result = $g_db->get_server_info();
if( version_compare( $t_result['version'], '1.0' ) >= 0 )
{ ... }
```

> **Note:** The information element is what is returned by PDO's GetAttribute for the `PDO::ATTR_SERVER_INFO` variable. This can be an array or a string for different database engines.

get_indexes

This function returns details of any indexes on the given database table

```
public function get_indexes($p_table);
```

The information returned is designed to be consistent across the 4 database backend engines.

The output of this function is an array of objects containing:

```
array['IndexName']->unique = true|false
array['IndexName']->primary = true|false
array['IndexName']->columns = array( column1, column2 );
```

get_columns

This function returns details of the columns on the given database table

```
public function get_indexes($p_table);
```

The information returned is designed to be consistent across the 4 database backend engines.

The output of this function is an array of objects containing:

```
array['TableName']->name = TableName
array['TableName']->not_null = true|false
array['TableName']->primary = true|false
array['TableName']->auto_increment = true|false
array['TableName']->binary = true|false
array['TableName']->unsigned = true|false
```

```
array['TableName']->has_default = true|false
array['TableName']->default_value = ....
array['TableName']->scale = ....
array['TableName']->type = field type **note: should make this portable type**
array['TableName']->max_length = numeric
```

->table_exists

MantisDatabaseDict Class

The purpose of the MantisDatabaseDict set of classes is to handle schema modification across the database engines we support.

The List of supported Database Engines can be found here.

The Schema Abstraction layer provides standard Data Types that can be used across the database engines, and the following functions to return SQL Code to manipulate the database schema :

- Database Manipulation
  - CreateDatabase
  - DropDatabase
- Table Manipulation
  - CreateTable
  - RenameTable [remove?]
  - DropTable
- Table Columns Manipulation
  - AddColumn
  - RenameColumn [remove?]
  - DropColumn
  - AlterColumn
- Index Manipulation
  - CreateIndex
  - DropIndex

The above functions return an array of SQL queries to run to achieve the desired result. These should be passed to the ExecuteSQLarray function. An example of the process to initialise the MantisDatabaseDict and create a database is shown below:

```
// 1: Get a valid MantisDatabase Object ($g_db) for a database conneciton
$g_db = MantisDatabase::get_driver_instance($g_db_type);
$t_result = $g_db->connect( null, $g_hostname, $g_username, $g_password, null, null );
$g_db->SetPrefixes( $g_prefix, $g_suffix );

// 2: Get a valid MantisDatabaseDict object for the correct database type to manipulate the schema
$g_dict = MantisDatabaseDict::get_driver_instance($g_db_type);

// 3: Call the Create Database function and store an array of SQL queries to create the database.
$sqlarray = $g_dict->CreateDatabase( $this->database_name );

// 4: Execute the Query Array. Note: this makes use of the database connection in step 1.
$ret = $g_dict->ExecuteSQLarray( $sqlarray );
```

Data Types

Data Type	Type Identifier	Oracle	Pgsql	MSSQL	Mysql	Notes
Variable Character	C(NN)		VARCHAR(NN)	NVARCHAR(NN)	VARCHAR(NN)	Limited to 255 Characters. **Note** can specify nn to set number of characters.
Largest Text	XL		TEXT	NVARCHAR(MAX)	LONGTEXT	

4-byte Integer	I		INTEGER	INTEGER	INTEGER	
2-byte Integer	I2		INT2	SMALLINT	SMALLINT	
Integer Boolean Field	L		SMALLINT	TINYINT	TINYINT	
Binary Files/Blob	B		BYTEA	VARBINARY(MAX)	LONGBLOB	
Varchar < 4000 charcters	X		VARCHAR(4000)	NVARCHAR(4000)	TEXT	Limited to 4000 Characters (oracle)
Legacy Datetime	T		TIMESTAMP	DATETIME	DATETIME	**DO NOT USE** This is included for legacy reasons

Database Manipulation

CreateDatabase

This function returns the SQL to create a new database for the given database driver. It is expected that plugins would not use this function and the core purpose of this function is to create the initial Mantis Database within the installer.

```
function CreateDatabase($p_database_name,$p_options=null);
```

`$p_database_name` is the name of the database to create.

`$p_options`, if set, is an array of options to pass to the database layer of specific driver settings.

> **Note:** I want to remove $p_options, but this may be required for our oracle layer to define a password for the database creation

DropDatabase

This function returns the SQL to drop a database for the given database driver. It is expected that this function will not be used. The existence of this function is to allow Unit Tests within Mantis to reset state.

```
function DropDatabase($p_database_name);
```

`$p_database_name` is the name of the database to remove.

Table Manipulation

CreateTable

This function returns the SQL to create a new table within the database. It requires a definition of the fields within the new table, as outlined below.

```
function CreateTable($p_table_name, $p_fields, $p_table_options=array())
```

`$p_table_name` is the name of the new database table to create. It is expected that the new database does not already exist. The table_exists functions can be used to check if the table already exists.

`$p_fields` is the definition of the database columns/fields

`$p_table_options`, if set, is an array of options to pass to the database layer of specific driver settings.

> **Note:** I want to remove $p_table_options if possible (assuming there's no reason to keep)

The definition of `$p_fields` uses a text format, with each table column being comma-delimited. The format of a field follows the following structure:

```
fieldname datatype(size) otheroptions
```

This leads to a definition of new column looking similar to the below:

```
FirstName   C(50) NOTNULL DEFAULT 'Mr or Ms'
```

This creates a new column called FirstName, that is a 50 character varchar() field, with a default value of "Mr or Ms".

The List of Data Types available are listed under the DataTypes section. The optional Size parameter represents the size of the field for example (50) characters above.

The `otheroptions` section allows the following keywords (case insensitive):

Keyword	Description
AUTOINCREMENT	Indicates the column should auto-increment. On some database engines, triggers will be used to emulate this functionality.
PRIMARY	Indicates that this column is part of a Primary Key field. Compound Keys are supported.
DEFAULT	Indicates the default value to use for this column
NOTNULL	Do not allow this column to contain null values

As an example, the definition of the `$p_fields` parameter the mantis config table is as shown below

```
$p_fields = "   config_id C(64) NOTNULL PRIMARY,
                project_id I DEFAULT '0' PRIMARY,
                user_id I DEFAULT '0' PRIMARY,
                access_reqd I DEFAULT '0',
                type I DEFAULT '90',
                value XL NOTNULL";
```

RenameTable

> **Note:** Remove: in favour of CreateTable, move data, drop table - given we aren't normally going to be renaming tables??

DropTable

This function returns the SQL to remove a table from the database.

```
function DropTable($p_table_name)
```

Note: This will delete any data contained within the table.

Table Columns Manipulation

AddColumn

This function returns the SQL to add a new column to the given database table

```
function AddColumn($p_table_name, $p_fields)
```

`$p_table_name` is the name of the table to add the column to

`$p_fields` is the definition of the field(s) to add. See CreateTable for details on the format of this parameter.

RenameColumn

> **Note:** Remove: in favour of CreateColumn, move data, DropColumn - given we aren't normally going to be renaming columns??

DropColumn

This function returns the SQL to remove one or more columns from a table in the database

```
function DropColumn($p_table_name, $p_fields)
```

`$p_table_name` is the name of the table to remove the column from

`$p_fields` is the list of column(s) to remove..

Note: I'm tempted to think this should be done **1** column at a time and not allow an array.

AlterColumn

This function returns the SQL to change the definition of a column in the database

```
function AlterColumn($p_table_name, $p_fields);
```

`$p_table_name` is the name of the table to add the column to

`$p_fields` is the definition of the field(s) to add. See CreateTable for details on the format of this parameter.

> Note: It is not possible to arbitary change from some data types to others e.g. converting from a binary blob to a boolean value may not be possible. For details of options that are available please see <@T>

Index Manipulation

CreateIndex

This function returns the SQL to add an index for the specific table/column

```
function CreateIndex($p_index_name, $p_table_name, $p_fields, $p_index_options = false)
```

`$p_index_name` is the name of the index to create

`$p_table_name` is the table name to add the index to

`$p_fields` is the list of fields which should be indexed.

`$p_index_options` is an array of options to be used when creating the index. Supported options are detailed below:

Option	Description
UNIQUE	Create a Unique index

DropIndex

```
function DropIndex ($p_index_name, $p_table_name = null)
```

This function is used to return the SQL to drop an named index. If a table name is specified, we will look for the index on the given table.