Database Access

Database Access is made available through the use of the Record component, which utilizes PHP's PDO data-access library. This section provides an overview of the Record component and its common uses.

Table of Contents

Data Management

Select

This section describes common select statements. Follow each example with **->fetch()** to retrieve 1 record, or **->fetchAll()** to retrieve all records, or follow with **->get()** to retrieve the SQL for that statement. You may also **echo $this->Record** to print a query without resetting the query.

All

```
// Produces SELECT * FROM `clients`
$this->Record->select()->from("clients");
```

Tuples

```
// Produces SELECT `id`,`id_code`,`user_id` FROM `clients`
$this->Record->select(array("id","id_code","user_id"))->from("clients");
```

Tuple Aliasing

```
// Produces SELECT `id`,`id_code` AS `key`,`user_id` FROM `clients`
$this->Record->select(array("id","id_code"=>"key","user_id"))->from("clients");

// Produces SELECT `id`, 'customer' AS `type` FROM `clients`
$this->Record->select(array("id"))->select(array('\'customer\''=> "type"), false)->from("clients");
```

Limit

```
// Produces SELECT * FROM `clients` LIMIT 20, 10
$this->Record->select()->from("clients")->limit(10, 20);
```

Order By

```
// Produces SELECT * FROM `clients` ORDER BY `clients`.`user_id` ASC
$this->Record->select()->from("clients")->order(array("clients.user_id"=>"asc"));
```

Where

```
// Produces SELECT * FROM `clients` WHERE `clients`.`id`='15'
$this->Record->select()->from("clients")->where("clients.id", "=", 15);
```

And Where

```
// Produces SELECT * FROM `clients` WHERE `user_id`='15' AND `id_code`='USR15'
$this->Record->select()->from("clients")->where("user_id", "=", 15)->where("id_code", "=", "USR15");
```

Or Where

```
// Produces SELECT * FROM `clients` WHERE `user_id`='15' OR `id_code`='USR15'
$this->Record->select()->from("clients")->where("user_id", "=", 15)->orWhere("id_code", "=", "USR15");
```

Where In

```
// Produces SELECT * FROM `clients` WHERE `id` IN ('1','2','3','4')
$this->Record->select()->from("clients")->where("id", "in", array(1,2,3,4));
```

Like

```
// Produces SELECT * FROM `clients` WHERE `id_code` LIKE 'USR15'
$this->Record->select()->from("clients")->like("id_code", "USR15");

// Produces SELECT * FROM `clients` WHERE `id_code` LIKE '%USR15%'
$this->Record->select()->from("clients")->like("id_code", "%USR15%");
```

Or Like

```
// Produces SELECT * FROM `clients` WHERE `user_id`='15' OR `id_code` LIKE '%USR15%'
$this->Record->select()->from("clients")->where("user_id", "=", 15)->orLike("id_code", "%USR15%");
```

Group By

```
// Produces SELECT * FROM `clients` WHERE `id_code` LIKE '%USR15%' GROUP BY `clients`.`user_id`, `clients`.
`id_code`
$this->Record->select()->from("clients")->like("id_code", "%USR15%")->group(array("clients.user_id", "clients.
id_code"));
```

Having

```
// Produces SELECT * FROM `clients` HAVING `primary_account_type` IS NOT NULL
$this->Record->select()->from("clients")->having("primary_account_type", "!=", null);

// Produces SELECT * FROM `clients` HAVING `primary_account_type` IS NOT NULL OR `id`>'0'
$this->Record->select()->from("clients")->having("primary_account_type", "!=", null)->orHaving("id", ">", 0);
```

Aggregate Functions

```
// Produces SELECT MAX(`id`) AS `largest_id` FROM `clients`
$this->Record->select(array("MAX(id)"=>"largest_id"))->from("clients");
```

Grouping Conditions

```
// Produces SELECT * FROM `clients` WHERE ((`id_code` LIKE '%USR15%' OR `id_code` LIKE '%USR18%') AND `name`
LIKE 'Firstname%')
$this->Record->select()->from("clients")->
    open()->
        open()->
            like("id_code", "%USR15%")->
            orLike("id_code", "%USR18%")->
        close()->
        like("name", "Firstname%")->
    close();
```

Joins

Inner Join

```
// Produces SELECT `clients`.* FROM `clients` INNER JOIN `contacts` ON `contacts`.`user_id`=`clients`.`user_id`
$this->Record->select(array("clients.*"))->from("clients")->innerJoin("contacts", "contacts.user_id", "=",
"clients.user_id", false);
```

Left Join

```
// Produces SELECT `clients`.* FROM `clients` LEFT JOIN `contacts` ON `contacts`.`user_id`=`clients`.`user_id`
$this->Record->select(array("clients.*"))->from("clients")->leftJoin("contacts", "contacts.user_id", "=",
"clients.user_id", false);
```

Right Join

```
// Produces SELECT `clients`.* FROM `clients` right JOIN `contacts` ON `contacts`.`user_id`=`clients`.`user_id`
$this->Record->select(array("clients.*"))->from("clients")->rightJoin("contacts", "contacts.user_id", "=",
"clients.user_id", false);
```

Cross Join

```
// Produces SELECT `clients`.* FROM `clients`, `contacts`
$this->Record->select(array("clients.*"))->from("clients")->from("contacts");

// Produces SELECT `clients`.* FROM `clients` JOIN `contacts`
$this->Record->select(array("clients.*"))->from("clients")->join("contacts");
```

Multiple conditions

```
// Produces SELECT `clients`.* FROM `clients` LEFT JOIN `contacts` ON `contacts`.`user_id`=`clients`.`user_id`
AND `contacts`.`id`!='0'
$this->Record->select(array("clients.*"))->from("clients")->on("contacts.id", "!=", 0)->leftJoin("contacts",
"contacts.user_id", "=", "clients.user_id", false);

// Produces SELECT `clients`.* FROM `clients` LEFT JOIN `contacts` ON `contacts`.`user_id`=`clients`.`user_id`
OR `contacts`.`id`!='0'
$this->Record->select(array("clients.*"))->from("clients")->orOn("contacts.id", "!=", 0)->leftJoin("contacts",
"contacts.user_id", "=", "clients.user_id", false);
```

Subqueries

All subqueries start first with the subquery. The idea is to construct the query from the inside out, and as each layer is added the subquery becomes part of the parent query.

```
$client_id = 12;

// Produces SELECT `clients`.*, `contacts`.* FROM `clients` INNER JOIN `contacts` ON `contacts`.
`client_id`=`clients`.`id` WHERE `clients`.`id`='12' AND `contacts`.`user_id`=`clients`.`user_id`
$sub_query_record = new Record();
$sub_query_record->select(array("clients.*","contacts.*"))->from("clients")->
    innerJoin("contacts", "contacts.client_id", "=", "clients.id", false)->
    where("clients.id", "=", $client_id)->
    where("contacts.user_id", "=", "clients.user_id", false);

// Build the SQL for the sub_query
$sub_query_sql = $sub_query_record->get();

// Fetch any values set from the subquery
$values = $sub_query_record->values;
```

Join on Subquery

```
// Produces SELECT `users`.* FROM `users` INNER JOIN (SELECT `clients`.*, `contacts`.* FROM
// `clients` INNER JOIN `contacts` ON `contacts`.`client_id`=`clients`.`id` WHERE `clients`.`id`='12' AND
`contacts`.`user_id`=`clients`.`user_id`)
// AS `temp` ON `temp`.`user_id`=`users`.`id`
$this->Record->select("users.*")->appendValues($values)->from("users")->innerJoin(array($sub_query_sql=>"
temp"), "temp.user_id", "=", "users.id", false);
```

Subquery as a Tuple

```
// Produces SELECT `temp`.`id` FROM (SELECT `clients`.*, `contacts`.* FROM
// `clients` INNER JOIN `contacts` ON `contacts`.`client_id`=`clients`.`id` WHERE `clients`.`id`='12' AND
`contacts`.`user_id`=`clients`.`user_id`)
// AS `temp` WHERE `temp`.`id`='125'
$this->Record->select("temp.id")->appendValues($values)->from(array($sub_query_sql=>"temp"))->where("temp.id","
=",125);
```

Subquery as a Column

```
// Produces SELECT `clients`.`id`, (SELECT `clients`.*, `contacts`.* FROM `clients` INNER JOIN `contacts` ON
`contacts`.`client_id`=`clients`.`id`
// WHERE `clients`.`id`='12' AND `contacts`.`user_id`=`clients`.`user_id`) AS `temp` FROM `clients` WHERE
`clients`.`id`='125'
$this->Record->select("clients.id")->appendValues($values)->select(array("(" . $sub_query_sql . ")"=>"temp"),
false)->from("clients")->where("clients.id","=",125);
```

Where not in Subquery

```
// Produces SELECT * FROM `clients` WHERE `id` NOT IN (SELECT `clients`.*, `contacts`.* FROM `clients` INNER
JOIN
// `contacts` ON `contacts`.`client_id`=`clients`.`id` WHERE `clients`.`id`='12' AND `contacts`.
`user_id`=`clients`.`user_id`)
$this->Record->select()->from("clients")->appendValues($values)->where("id", "notin", array($sub_query_sql),
false);
```

Miscellaneous Select Statements

Result Count

```
$this->Record->select()->from("clients")->numResults();
```

Last Insert ID

```
$this->Record->lastInsertId();
```

Custom SQL

```
// Select with single parameter
$this->Record->query("SELECT `clients`.* FROM `clients` WHERE `clients`.`id`=?", 12);

// Select with multiple parameters
$this->Record->query("SELECT `clients`.* FROM `clients` WHERE `clients`.`id`=? OR `clients`.`id`=?", 12, 15);

// Select with array of parameters
$this->Record->query("SELECT `clients`.* FROM `clients` WHERE `clients`.`id`=? OR `clients`.`id`=?", array(12,
15));
```

Insert

Simple Insert

```
// Executes INSERT INTO `clients` (`user_id`,`id_code`) VALUES ('15','USR15')
$this->Record->insert("clients", array('user_id' => 15,'id_code' => "USR15"));

// Executes INSERT INTO `clients` (`user_id`,`id_code`) VALUES ('15','USR15')
$this->Record->set("id_code","USR15")->set("user_id", 15)->insert("clients");
```

Insert with Filter

```
// Executes INSERT INTO `clients` (`user_id`,`id_code`) VALUES ('15','USR15')
$this->Record->insert("clients", array('user_id' => 15,'id_code' => "USR15", 'first_name' => "First",
'last_name' => "Last"), array("user_id","id_code"));
```

On Duplicate

```
// Executes INSERT INTO `clients` (`user_id`,`id_code`) VALUES ('15','USR15') ON DUPLICATE KEY UPDATE
`id_code`='USR15'
$this->Record->duplicate("id_code", "=", "USR15")->insert("clients", array('user_id' => 15,'id_code' =>
"USR15"));
```

From Subquery

```
$sub_query_record = new Record();
$sub_query_record->select(array("clients.id"))->from("clients")->where("clients.id", "=", 12);

// Executes INSERT INTO `some_table` (`value`) (SELECT `clients`.`id` FROM `clients` WHERE `clients`.`id`='12')
$this->Record->insert("some_table", array('value'=>$sub_query_record));
```

Update

Simple Update

```
// Executes UPDATE `clients` SET `user_id`='15', `id_code`='USR15'
$this->Record->update("clients", array('user_id' => 15, 'id_code' => "USR15"));

// Executes UPDATE `clients` SET `id_code`='USR15' WHERE `user_id`='15'
$this->Record->where("user_id", "=", 15)->update("clients", array("id_code"=>"USR15"));
```

Update with Filter

```
// Executes UPDATE `clients` SET `user_id`='15', `id_code`='USR15'
$this->Record->update("clients", array('user_id' => 15,'id_code' => "USR15", 'first_name' => "First",
'last_name' => "Last"), array("user_id","id_code"));
```

Update with Increment

```
// Executes UPDATE `clients` SET `user_id`=user_id+3 WHERE `user_id`='15'
$this->Record->set("user_id","user_id+3", false, false)->where("user_id", "=", 15)->update("clients");
```

Delete

Simple Delete

```
// Executes DELETE FROM `clients`
$this->Record->from("clients")->delete();
```

Multi-delete

```
// Executes DELETE `clients`.*, `contacts`.* FROM `clients` INNER JOIN `contacts` ON `contacts`.
`client_id`=`clients`.`id` WHERE `clients`.`id`='15'
$this->Record->from("clients")->innerJoin("contacts", "contacts.client_id", "=", "clients.id")->where("clients.
id", "=", 15)->delete(array("clients.*","contacts.*"));
```

Table Management

Create

```
// Executes CREATE TABLE `feed_reader_feeds` (
// `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
// `url` varchar(255) NOT NULL,
// `updated` datetime DEFAULT NULL,
// PRIMARY KEY (`id`),
// UNIQUE KEY `url` (`url`)
// ) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
$this->Record->
     setField("id", array('type' => "int",'size' => 10,'unsigned' => true,'auto_increment' => true))->
     setField("url", array('type' => "varchar", 'size' => 255))->
     setField("updated", array('type' => "datetime", 'is_null' => true, 'default' => null))->
     setKey(array("id"), "primary")->
     setKey(array("url"), "unique")->
     create("feed_reader_feeds");
```

Alter

```
// Executes ALTER TABLE `feed_reader_feeds`
// ADD `title` varchar(255) NOT NULL,
// DROP UNIQUE KEY `url`
$this->Record->
     setField("title", array('type' => "varchar", 'size' => 255))->
     setKey(array("url"), "unique", null, false)->
     alter("feed_reader_feeds");
```

Truncate

```
// Executes TRUNCATE TABLE `feed_reader_feeds`
$this->Record->truncate("feed_reader_feeds");
```

Drop

```
// Executes DROP TABLE `feed_reader_feeds`
$this->Record->drop("feed_reader_feeds");
```

Transactions

Transactions offer a method of executing queries in series, with the ability to roll back to a previous state if an error occurs. Any database error will throw a PDOException.

> ⚠ **Transactions must be supported by the database engine**
>
> Blesta uses the InnoDB engine for all of its tables, and you are encouraged to do the same. Transactions will fail on database engines that do not support transactions.

Using Transactions

A Model

```
try {
    // Begin transaction
    $this->Record->begin();

    // Run queries...

    // Commit queries
    $this->Record->commit();
}
catch (PDOException $e) {
    // Rollback the queries, error occurred.
    $this->Record->rollback();
}
```