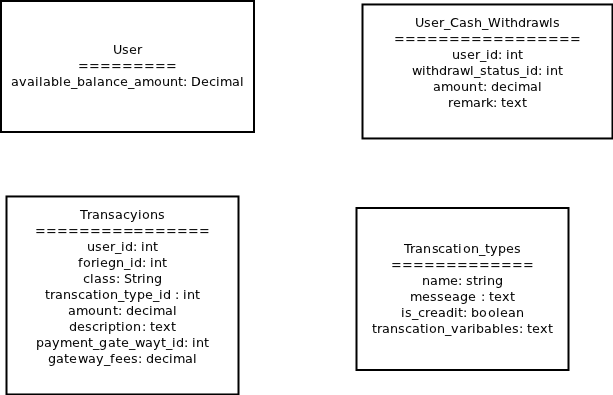
Double Entry Lager



**How the system is storing transactions?**

Here the system is storing everything as a unique transactions. Not separate withdraw or deposit.

TransactionType is determined by the flag is\_credit . While listing all the transactions

|  |  |
| --- | --- |
| Debit | Credit |
| 5.0 | 0 |
| 0 | 5.0 |

is\_credit is used to determine the type of the transaction.

**What is user available amount?**

This what we call account.amount in Monopoly. After buy or selling this gets updated.

**How the transaction is related with other models?**

There is a polymorhic association between transaction and other models like deal, second\_user etc.

This way anything can belongs to a transactions.

**What is happening while admin first deposit the account?**

The system logs this statically that admin has deposit some money. That is done by the the Second user relationship. Second user is simple mock user model … using same table in the background. This way system can track which user has deposit the amount to a User account.

For HB,

There is no way we can use any part of this code, this is way to messy … we only can use the idea.

**Models**

<?php

class Transaction extends AppModel

{

var $name = 'Transaction';

var $actsAs = array(

'Polymorphic' => array(

'classField' => 'class',

'foreignKey' => 'foreign\_id',

)

);

//$validate set in \_\_construct for multi-language support

//The Associations below have been created with all possible keys, those that are not needed can be removed

var $belongsTo = array(

'User' => array(

'className' => 'User',

'foreignKey' => 'user\_id',

'conditions' => '',

'fields' => '',

'order' => '',

) ,

'TransactionType' => array(

'className' => 'TransactionType',

'foreignKey' => 'transaction\_type\_id',

'conditions' => '',

'fields' => '',

'order' => '',

) ,

'PaymentGateway' => array(

'className' => 'PaymentGateway',

'foreignKey' => 'payment\_gateway\_id',

'conditions' => '',

'fields' => '',

'order' => '',

) ,

'SecondUser' => array(

'className' => 'SecondUser',

'foreignKey' => 'foreign\_id',

'conditions' => '',

'fields' => '',

'order' => '',

) ,

'GiftUser' => array(

'className' => 'GiftUser',

'foreignKey' => 'foreign\_id',

'conditions' => '',

'fields' => '',

'order' => '',

) ,

'DealUser' => array(

'className' => 'DealUser',

'foreignKey' => 'foreign\_id',

'conditions' => '',

'fields' => '',

'order' => '',

) ,

'Deal' => array(

'className' => 'Deal',

'foreignKey' => 'foreign\_id',

'conditions' => '',

'fields' => '',

'order' => '',

)

);

function \_\_construct($id = false, $table = null, $ds = null)

{

parent::\_\_construct($id, $table, $ds);

$this->validate = array(

'user\_id' => array(

'rule' => 'numeric',

'allowEmpty' => false,

'message' => \_\_l('Required')

) ,

'amount' => array(

'rule2' => array(

'rule' => array(

'comparison',

'>',

0

) ,

'allowEmpty' => false,

'message' => \_\_l('Should be greater than 0')

) ,

'rule1' => array(

'rule' => 'numeric',

'allowEmpty' => false,

'message' => \_\_l('Required')

) ,

)

);

}

function log($data)

{

if (!empty($data)) {

$this->create();

$this->save($data);

return $this->getLastInsertId();

}

return false;

}

}

?>