projekt kivitendo - Fehler #265

Kontoauszug verbuchen bei negativer Kreditorenbuchung wird das Vorzeichen bei Zahlung umgedreht

07.06.2017 10:28 - Jan Büren

Status:	Erledigt	**Beginn:**	07.06.2017
Priorität:	Normal	**Abgabedatum:**	
Zugewiesen an:		**% erledigt:**	0%
Kategorie:		**Geschätzter Aufwand:**	0.00 Stunde
Zielversion:	3.5	**Aufgewendete Zeit:**	0.00 Stunde

Beschreibung

Wie https://redmine.kivitendo-premium.de/issues/243 nur diesmal für Kreditorenbuchugen (Einkaufsrechnung ggf. auch).

Historie

#1 - 13.06.2017 12:59 - Jan Büren

- Zielversion wurde auf 3.5 gesetzt

#2 - 30.06.2017 13:01 - Jan Büren

- Status wurde von Neu zu Abgewiesen geändert

Konnte ich selber mit der aktuellsten Version nicht mehr nachvollziehen.
Testfälle hierfür sind soweit auch sauber.

Lediglich der Hinweis bei der Punktevergabe: "wrong sign" könnte noch überarbeitet werden

```
   #check sign
+  if ( $invoice->is_sales && $invoice->amount > 0 && $self->amount < 0 ) {
     $agreement += $points{wrong_sign};
     $rule_matches .= 'wrong_sign(' . $points{'wrong_sign'} . ') ';
   };
+   if ( ! $invoice->is_sales && $invoice->amount < 0 && $self->amount > 0 ) {
     $agreement += $points{wrong_sign};
     $rule_matches .= 'wrong_sign(' . $points{'wrong_sign'} . ') ';
   };
```

Damit kann man die defensive Punktevergabe von -1 auch höher setzen.

#3 - 22.01.2018 15:22 - Jan Büren

- Status wurde von Abgewiesen zu Neu geändert

Ok, konnte ich doch reproduzieren.

Mit Commit #b6f37661 wurde das Verhalten für negative Einkaufs- oder Kreditorenbelege verbessert, allerdings NUR wenn der Rechnungsbetrag exakt mit dem Kontoauszugsbetrag übereinstimmt.

```
+         } elsif (!$invoice->is_sales && $invoice->invoice_type eq 'ap_transaction' ) {
+           # $invoice->open_amount may be negative for ap_transaction but may be positiv for negativ ap_transa
ction
+           # if $invoice->open_amount is negative $bank_transaction->amount is positve
+           # if $invoice->open_amount is positive $bank_transaction->amount is negative
+           # but amount of transaction is for both positive
+           $amount_of_transaction *= -1 if $invoice->open_amount == - $amount_of_transaction;
+         }
```

Wie in #242 schon andiskutiert, kann ich das Datenmodell nicht nachvollziehen, dass innerhalb Perls noch die Vorzeichen für die Zahlungen bestimmt werden müssen.

Im speziellen Fall wird das Vorzeichen korrekt gesetzt wenn der offene Betrag der Rechnung auch der wirklichen Summe der Transaktion entspricht. Das passt dann aber für den Fall Lieferantengutschrift mit Skonto nicht und die Buchungen gehen in die falsche Richtung.

Vom Objekt-Modell wäre es sinnvoll noch die Objekte Einkaufs-Lieferantengutschrift und Kreditorengutschrift oder sowas zu haben.
Definierbar wie folgt:

SL/DB/PurchaseInvoice.pm

```
sub invoice_type {
  my ($self) = @_;

+  return 'ap_transaction'             if !$self->invoice && $self->amount < 0;
+  return 'credit_note_ap_transaction' if !$self->invoice && $self->amount > 0;
+  return 'purchase_credit_note'       if  $self->invoice && $self->amount > 0;
  return 'purchase_invoice';
}
```

Ansonsten kann ich die entsprechende Stelle auf zwei Zustände runterbrechen.

Entweder es handelt sich um den Fall Gutschrift (egal ob im Verkauf oder Einkauf), dann ist $invoice->amount < 0, da dieser Wert aus ap oder ar kommt.

```
 if ( $invoice->is_sales && $invoice->amount < 0 ) {

          $amount_of_transaction *= -1;
        # this catches negative purchase invoices and ap_transactions
 } elsif (!$invoice->is_sales && $invoice->amount < 0) {

          $amount_of_transaction *= -1;
 }
```

Das kann natürlich weiter eingedampft werden.

Grundannahme ist, dass dieses Konstrukt optimierungswürdig ist:

```
        } elsif (!$invoice->is_sales && $invoice->invoice_type =~ m/ap_transaction|purchase_invoice/) {
          # $invoice->open_amount may be negative for ap_transaction but may be positiv for negativ ap_transac
tion
          # if $invoice->open_amount is negative $bank_transaction->amount is positve
          # if $invoice->open_amount is positive $bank_transaction->amount is negative
          # but amount of transaction is for both positive
          $amount_of_transaction *= -1 if $invoice->open_amount == - $amount_of_transaction;
        }
```

Ich hab spontan keinen Testfall dagegen erzeugen können, gehe aber davon aus, dass nur der Fall Bankzahlungen gleich exakte Rechnungssumme abgedeckt ist.
Lesbarere wäre es ferner ohne das postif.

#4 - 23.01.2018 14:05 - Jan Büren

Ok, das postif ist deshalb reingerutscht weil der Payment Helper im späteren Verlauf bei dem Fall Skontobetrag das Vorzeichen dreht.

Hier mein Patch-Vorschlag:

```
From a96954ecc8f48860209cae23362bcdf7ee2808bc Mon Sep 17 00:00:00 2001
From: =?UTF-8?q?Jan=20B=C3=BCren?= <jan@kivitendo-premium.de>
Date: Tue, 23 Jan 2018 13:48:13 +0100
Subject: [PATCH 1/1] Kontoauszug verbuchen - Teil oder skontobezahlte
 Lieferantengutschriften richtig behandeln
MIME-Version: 1.0
Content-Type: text/plain; charset=UTF-8
Content-Transfer-Encoding: 8bit

Behebt #265. Der ursprüngliche Fix konnte nur komplett
bezahlte Lieferantengutschriften verbuchen - Dies ist jetzt
im Banktransaction Controller auch auf Teilzahlungen erweitert.
Zusätzlich musste hierfür noch der Payment-Helper erweitert werden,
der auch über den Typ des Belegs Vermutungen über die Art des Bezahlens macht.
Das Punktesystem für die Vorschlagsliste bei Kontoauszug verbuchen, passt
somit dann auch nicht mehr und ist damit auch konsequent geändert.
---
 SL/Controller/BankTransaction.pm | 28 ++++++++++++----------------
 SL/DB/BankTransaction.pm         |  3 ++-
 SL/DB/Helper/Payment.pm          | 15 ++++++++++-----
 3 files changed, 24 insertions(+), 22 deletions(-)

diff --git a/SL/Controller/BankTransaction.pm b/SL/Controller/BankTransaction.pm
index fcea622..921af32 100644
--- a/SL/Controller/BankTransaction.pm
```

```
+++ b/SL/Controller/BankTransaction.pm
@@ -652,22 +652,18 @@ sub save_single_bank_transaction {
                                memo         => $memo,
                                transdate    => $bank_transaction->transdate->to_kivitendo);
       } else {
-        # use the whole amount of the bank transaction for the invoice, overpay the invoice if necessary
-
-        # this catches credit_notes and negative sales invoices
-        if ( $invoice->is_sales && $invoice->amount < 0 ) {
-          # $invoice->open_amount     is negative for credit_notes
-          # $bank_transaction->amount is negative for outgoing transactions
-          # so $amount_of_transaction is negative but needs positive
-          $amount_of_transaction *= -1;
-
-        } elsif (!$invoice->is_sales && $invoice->invoice_type =~ m/ap_transaction|purchase_invoice/) {
-          # $invoice->open_amount may be negative for ap_transaction but may be positiv for negativ ap_transa
ction
-          # if $invoice->open_amount is negative $bank_transaction->amount is positve
-          # if $invoice->open_amount is positive $bank_transaction->amount is negative
-          # but amount of transaction is for both positive
-          $amount_of_transaction *= -1 if $invoice->open_amount == - $amount_of_transaction;
-        }
+      # use the whole amount of the bank transaction for the invoice, overpay the invoice if necessary
+
+        # $invoice->open_amount     is negative for credit_notes
+        # $bank_transaction->amount is negative for outgoing transactions
+        # so $amount_of_transaction is negative but needs positive
+        # $invoice->open_amount may be negative for ap_transaction but may be positiv for negative ap_transac
tion
+        # if $invoice->open_amount is negative $bank_transaction->amount is positve
+        # if $invoice->open_amount is positive $bank_transaction->amount is negative
+        # but amount of transaction is for both positive
+
+        $amount_of_transaction *= -1 if ($invoice->amount < 0);
+

         my $overpaid_amount = $amount_of_transaction - $invoice->open_amount;
         $invoice->pay_invoice(chart_id     => $bank_transaction->local_bank_account->chart_id,
diff --git a/SL/DB/BankTransaction.pm b/SL/DB/BankTransaction.pm
index 55bf9b9..486c160 100644
--- a/SL/DB/BankTransaction.pm
+++ b/SL/DB/BankTransaction.pm
@@ -144,7 +144,8 @@ sub get_agreement_with_invoice {
     $agreement += $points{wrong_sign};
     $rule_matches .= 'wrong_sign(' . $points{'wrong_sign'} . ') ';
   }
-  if ( ! $invoice->is_sales && $self->amount > 0 ) {
+  # better guess: - we may have negative purchase invoices or credit bookings
+  if ( ! $invoice->is_sales && $self->amount > 0 && $invoice->amount > 0 ) {
     $agreement += $points{wrong_sign};
     $rule_matches .= 'wrong_sign(' . $points{'wrong_sign'} . ') ';
   }
diff --git a/SL/DB/Helper/Payment.pm b/SL/DB/Helper/Payment.pm
index ae7fa53..21d056f 100644
--- a/SL/DB/Helper/Payment.pm
+++ b/SL/DB/Helper/Payment.pm
@@ -129,7 +129,8 @@ sub pay_invoice {

     # as long as there is no automatic tax, payments are always booked with
     # taxkey 0
-
+    # TODO rename: should be payment_amount or similiar
+    my $amount;
     unless ( $params{payment_type} eq 'difference_as_skonto' ) {
       # cases with_skonto_pt and without_skonto

@@ -142,7 +143,7 @@ sub pay_invoice {
       # bank account and AR/AP
       $paid_amount += $pay_amount * $exchangerate;

-      my $amount = (-1 * $pay_amount) * $mult;
+      $amount = (-1 * $pay_amount) * $mult;

       # total amount against bank, do we already know this by now?
@@ -220,11 +221,15 @@ sub pay_invoice {
```

```
       foreach my $skonto_booking ( @skonto_bookings ) {
         next unless $skonto_booking->{'chart_id'};
         next unless $skonto_booking->{'skonto_amount'} != 0;
-        my $amount = -1 * $skonto_booking->{skonto_amount};
+        my $skonto_amount = $skonto_booking->{skonto_amount};
+        # we cannot safely know if we have a credit or a debit case, but
+        # using the same sign as the payment booking should always be correct.
+        $skonto_amount *= -1 unless ($amount < 0 == $skonto_amount < 0);
+
         $new_acc_trans = SL::DB::AccTransaction->new(trans_id   => $self->id,
                                                      chart_id   => $skonto_booking->{'chart_id'},
                                                      chart_link => SL::DB::Manager::Chart->find_by(id => $sko
nto_booking->{'chart_id'})->link,
-                                                     amount     => $amount * $mult,
+                                                     amount     => $skonto_amount,
                                                      transdate  => $transdate_obj,
                                                      source     => $params{source},
                                                      taxkey     => 0,
@@ -302,7 +307,7 @@ sub pay_invoice {

       if ($datev->errors) {
         # this exception should be caught by with_transaction, which handles the rollback
-        die join "\n", $::locale->text('DATEV check returned errors:'), $datev->errors;
+        die join "\n", $::locale->text('DATEV check returned errors:' . Dumper($datev)), $datev->errors;
       }
     }

--
1.9.1
```

#5 - 20.11.2018 09:30 - Jan Büren

- Status wurde von Neu zu Erledigt geändert