projekt kivitendo - Fehler #342

inkonsistente Werte in »invoice.sellprice«

06.02.2018 13:16 - G. Richardson

Status:	Neu	**Beginn:**	06.02.2018
Priorität:	Normal	**Abgabedatum:**	
Zugewiesen an:		**% erledigt:**	0%
Kategorie:		**Geschätzter Aufwand:**	0.00 Stunde
Zielversion:		**Aufgewendete Zeit:**	0.00 Stunde

Beschreibung

zur Vorgeschichte, siehe Develliste:

Eines der Ziele ist ja eine einheitliche Behandlung von sellprice, die auch mit den Rose-Objekten funktioniert:

sellprice soll sein:

- angezeigter / eingegebener Stückpreis an Oberfläche
- ohne Rabatt in Belegwährung
- brutto oder netto, je nach Steuereinstellung des Belegs (taxincluded)

Mosus ursprünglicher Ansatz:

- angezeigter Stückpreis (was Benutzer eingegeben hat)
- angezeigte Zeilensumme (Rechnungswährung)
- interne Zeilensumme (in acc_trans-Währung)
- interner Stückpreis (kalkulatorisch, ungerundet)

Mein Ansatz:

Beim Speichern:

- erst in Belegwährung die Zeilensumme bestimmen (als Netto oder Brutto, je nach taxincluded)
- Rabatt wird auf die Zeilensumme angewandt und Zeilensumme gerundet, und zusätzlich absoluter Rabatt bestimmt
- je nach taxincluded wird die resultierende Netto- oder Bruttozeilensumme bestimmt, und daraus ergibt sich auch ein absoluter Steuerbetrag

Entscheidend ist eine saubere und einheitliche Berechnung der Zeilensummen, die man sowohl im Backend als auch im Frontend verwenden kann.

Ich würde folgende Informationen in den items speichern:

- angezeigter Stückpreis - wie Mosu, sellprice
- Zeilensumme netto in Rechnungswährung
- Zeilensumme brutto in Rechnungswährung
- interner Stückpreis pro Basiseinheit (kalkulatorisch, ungerundet)
- (absoluter Rabattbetrag, als Netto oder Brutto je nach taxincluded)

Beim Speichern von Belegen wird für jede Zeile folgende Information benötigt, um an alle notwendigen Zeilensummen zu kommen:

- qty
- unit
- sellprice
- price_factor
- discount
- (exchangerate)
- taxincluded
- taxrate

Was mir noch nicht klar ist: könnte man auf die interne Zeilensumme von Mosu komplett verzichten und diese bei Bedarf dynamisch berechnen? Oder sollte die direkt mitberechnet und ebenfalls gespeichert werden, weil eventuell Rundungsüberträge für die acc_trans mit berücksichtigt werden müssen?

Also
round(Zeilensumme * $exchangerate,2)
statt einmalig ermittelter interner Zeilensumme
:= round(angezeigter Stückpreis * qty / (price_factor || 1) * (1-discount) / (tax_included ? (1+tax_rate) : 1)

Ich habe zum Probieren eine Datenbankfunktion geschrieben (siehe Anhang), um die Werte zu bestimmen, mit folgenden Parametern:

glinetotal(qty, sellprice, price_factor, discount, exchangerate, taxincluded, taxrate)

Unit habe ich noch nicht berücksichtigt. Hier mal zwei Beispiele für den gleichen Sachverhalt, Menge 1000, Nettopreis 4.5, Rabatt 5%, als Steuer exkl. und inkl. Bei inklusive gebe ich als sellprice den Bruttoeinzelpreis ohne Rabatt ein, bei exklusive den Nettoeinzelpreis ohne Rabatt:

```
select * from glinetotal(1000, 4.5::numeric, 1, 0.05, 1, 'f', 0.19);
 --  linetotal_net             | 4275.00000
 --  linetotal_gross           | 5087.25000
 --  linetotal_tax_amount      |  812.25000
 --  linetotal_discount_amount |  225.00000
 --  sellprice_net             |    4.27500
```

```
select * from glinetotal(1000, (4.5*1.19)::numeric, 1, 0.05, 1, 't', 0.19);
 --  linetotal_net             | 4275.00
 --  linetotal_gross           | 5087.25000
 --  linetotal_tax_amount      |  812.25000
 --  linetotal_discount_amount |  267.75000
 --  sellprice_net             |    4.27500
```

Ich würde alle diese Zahlen außer linetotal_tax_amount explizit mit in items speichern.

Ein Vorteil der Datenbankfunktion: man kann bei bestehenden orderitems schon mal schauen, welche linetotals da rauskommen, zumindest bei den Belegen, die als Steuer exklusive gespeichert sind, und wo man die Parameter direkt an die Funktion übergeben kann (bis auf den Steuersatz, der ist umständlicher zu ermitteln). Beispiel, mit anderen Zahlen, nur als sql-Syntax-Beispiel:

```
select ois.qty,
           ois.sellprice,
           ois.discount,
           (glinetotal(ois.qty, ois.sellprice, ois.price_factor, ois.discount, 1, oe.taxincluded,
0.19)).*
      from orderitems ois
 left join oe on (oe.id = ois.trans_id);
```

```
qty | sellprice | discount  | linetotal_net | linetotal_gross | linetotal_tax_amount | linetotal_d
iscount_amount | sellprice_net
 ----+-----------+-----------+---------------+-----------------+----------------------+-----------
----------------+---------------
   1 |   9.60000 |      0.25 |       7.20000 |         8.57000 |              1.37000 |
        2.40000 |       7.20000
```

Warum absoluten Rabattbetrag separat speichern? Ähnlich wie sellprice ist discount der Betrag, der an der Oberfläche angezeigt wird/vom Benutzer vorgegeben wird.

Ich habe in einem Kundenprojekt den Fall, daß selbst nach Rabatt immer "runde" Einzelpreise gewünscht werden, also z.B. 3,50€, 3,75€, 4,00€, ...

Wenn man x% Rabatt gibt, kann es sein, daß sich bei manchen Artikeln der Preis gar nicht ändert, oder bei anderen Artikeln der effektive Rabatt höher oder niedriger als x% ist. Möchte ich den effektiven Rabattprozentwert wissen kann ich den aus den einmal bestimmten absoluten Zeilenrabatt berechnen (z.B. 4.67%), aber weiß, daß der Benutzer ursprünglich 5% eingeben hat (gespeichert in discount). Wie sellprice wird discount aber nach der Speicherung nur zum Anzeigen, nie wieder zum Rechnen benutzt.

Beispiel Berechnung des effektiven Nettorabatts aus dem Beispiel oben:
225 * 100 / (4275+225) = 5.00000

Dateien

glinetotal_0_1.sql	2,28 KB	06.02.2018	G. Richardson