

## Documentation Cashregister Sales.

Database cashregister:

```
cashregister=# \d
```

List of relations			
Schema	Name	Type	Owner
public	accounts	table	postgres
public	articles	table	postgres
public	buttons	table	postgres
public	loss	table	postgres
public	params	table	postgres
public	payments	table	postgres
public	sales	table	postgres

(7 rows)

Table "public.accounts"				
Column	Type	Collation	Nullable	Default
barcodeID	character varying(8)		not null	
firstname	character varying(20)			''::character varying
lastname	character varying(30)			''::character varying
access	integer			1
callname	character varying(20)			''::character varying

Indexes:

"barcodeID\_pkey" PRIMARY KEY, btree ("barcodeID")

```
cashregister=# \d articles
```

Table "public.articles"				
Column	Type	Collation	Nullable	Default
barcode	character varying(13)		not null	
description	character varying(50)			''::character varying
item_price	double precision			0
item_stock	double precision			0
item_unit	character varying(6)			''::character varying
minimum_stock	double precision			0
order_size	double precision			0
location_warehouse	character varying(8)			''::character varying
article_group	character varying(40)			''::character varying
thumbnail	character varying(50)			''::character varying
category	integer			0
order_balance	double precision			0
order_status	boolean			true
mutation_date	character varying(10)			''::character varying
annual_consumption_1	double precision			0
annual_consumption_2	double precision			0
VAT	character varying(4)			'high'::character varying

Indexes:

"barcode\_pkey" PRIMARY KEY, btree (barcode)

"barcode\_idx" btree (barcode)

cashregister=# \d buttons

Table "public.buttons"				
Column	Type	Collation	Nullable	Default
buttonID	integer		not null	
buttontext	character varying(30)			''::character varying
barcode	character varying(13)		not null	''::character varying

Indexes:

"buttonID\_pkey" PRIMARY KEY, btree ("buttonID")

cashregister=# \d loss

Table "public.loss"				
Column	Type	Collation	Nullable	Default
lossID	integer		not null	
number	double precision			0
category	character varying(22)			''::character varying
bookdate	character varying(10)			
barcode	character varying(13)			

Indexes:

"lossID\_pkey" PRIMARY KEY, btree ("lossID")

"fki\_barcode\_fkey" btree (barcode)

Foreign-key constraints:

"barcode\_fkey" FOREIGN KEY (barcode) REFERENCES articles(barcode)

cashregister=# \d params

Table "public.params"				
Column	Type	Collation	Nullable	Default
paramID	integer		not null	
item	character varying(20)			''::character varying
value	double precision			0
buttongroup	character varying(30)			''::character varying

Indexes:

"paramID\_pkey" PRIMARY KEY, btree ("paramID")

cashregister=# \d payments

Table "public.payments"				
Column	Type	Collation	Nullable	Default
payID	integer		not null	
kind	character varying(25)			''::character varying
amount	double precision			0
bookdate	character varying(10)			''::character varying
paydate	character varying(10)			''::character varying
instance	character varying(25)			''::character varying
accountnumber	character varying(25)			''::character varying
ovorderID	integer			0

Indexes:

"payID\_pkey" PRIMARY KEY, btree ("payID")

cashregister=# \d sales

Table "public.sales"				
Column	Type	Collation	Nullable	Default
ID	integer		not null	
receiptnumber	integer			0
barcode	character varying(13)			''::character varying
description	character varying(40)			''::character varying
number	double precision			0
item_price	double precision			0
sub_total	double precision			0
sub_vat	double precision			0
callname	character varying(20)			''::character varying
mutation_date	character varying(10)			''::character varying

Indexes:

"ID\_pkey" PRIMARY KEY, btree ("ID")

### Screenshot Cashregister frontend:

Cashregister Sales

Articlenumber	Description	Number	Item_price	Subtotal	VAT
8002000001119	Moederbord MSI Z97 U3 Plus	1	108.85	108.85	22.86
8002000001843	JetFlash 8GB USB Stick	1	7.60	7.60	1.60
8002000001591	Behuizing Midi Tower Corsair 450D	1	109.90	109.90	23.08
8002000002406	Spruitstuk RVS warmperswerk	1	18.34	18.34	3.85
8002000002512	Rvs Badkraanhuis warmperswerk	1	123.45	123.45	25.92
8002000005124	Houtdraadbout,thv 4.6,20x100mm,DIN571	1	0.94	0.94	0.20
8002000378082	Oxheart Cabbage	4	2.34	9.36	0.84
8002000379485	Orange	2	0.55	1.10	0.10
8002000378822	Kaiser Bun	4	0.33	1.32	0.12
8002000378945	Italian Bol	4	0.50	2.00	0.18
8002000379003	Baguette	2	1.02	2.04	0.18
8002000379973	Wine White glas	2	4.50	9.00	1.89
8002000380238	Coca Cola light glas	2	2.95	5.90	1.24
8002000001843	JetFlash 8GB USB Stick	1	7.60	7.60	1.60
8002000002406	Spruitstuk RVS warmperswerk	7	18.34	128.38	26.96
8002000005124	Houtdraadbout,thv 4.6,20x100mm,DIN571	1	0.94	0.94	0.20
Total incl. VAT				732.41	151.91

Error message: 3 in stock!

Administration

Vege tables	White Cabbage	Red Cabbage	Cauli Flower	Oxheart Cabbage	Avocado	Zucchini	Eggplant	Bell pepper Red	Bell pepper Green	Barcodescan	8002000002512
Cucumber	Basil	Chives								Number	3
										Information	Printing
										Exit	Next Customer

© 2020 all rights reserved dj.jansen@casema.nl

### General remarks by the used database items.

The items are not specific for retail, nor amounts and prices are realistic and for becoming real values the data should be applied to a working company and the data should be dynamic.

As a result the data looks sometimes unreal.



## CashRegister system.

The system detects if a logon barcode or a product barcode is scanned.

When no logon is established processing is blocked.

The red sign in the message bar below the display is showed.

By valid login the message bar show a green sign.

The logon is invisible with scanning.

With the logon from barcode accesslevel 3 a Administration button is activated. With this button and submenu accounts it is possible to generate other barcodes for logon purposes.

The barcodefield will be generated as 7 random digits by the program. (The 8th digit is a check number). The program will check and correct for duplicates. Access is default set on level 1, change if desired. The barcodelabel is saved in folder . /Barcodes/Accounts/

The first name, last name and callname of the employee must be inserted.

The callname field will be printed on the saleslip, it's also saved in the sales table.

When the account switch from level 1 to level 2 or level 3 or back the orderlines remains, so it's possible to book return goods with accepted account. If the account switches back the spinbox is reset to it's original state. When the login employee logs the barcode a second time, the employee is logged out.

When another employee logs his barcode, the logon is switched towards this employee.

Accesslevel 1. Normal operation. (No administration and no plusminbutton visible)

Accesslevel 2 . Expose an checkable button  $\pm$  for return products. The spinrange from the spinbox changes from 1, 99 to  $-1, -99$  with button checked.

Accesslevel 3. Expose the Administration button. Pressing this button reveals a combobox with 7 menulines. The items are:

Accounts

Submenu: New account  
View / Change accounts

Articles

Submenu: Insert new articles,  
View / Change articles  
Imports new articles  
Imports price-changes articles  
Imports expired articles  
View imports new articles  
View imports price-changes articles  
View imports expired articles  
Booking loss articles  
View loss articles

Sales

View

Payments

View / Pay.

Purchase

Submenu: Collecting purchases  
View purchases  
Printing purchases  
Processing deliveries  
View deliveries  
Printing deliveries

Buttons

Submenu : Buttons new barcode

Buttons existing barcode

Parameters

View / Change

## Articles

The table articles holds a column barcode (String 13 positions).

By inserting a new article a ean 13 barcode is generated. The first 2 numbers is the country code. The next 5 numbers are the company numbers, the following 5 numbers is the product number and the last number is a validity check number.

In this module is a image saved of the barcode in the folder ./Barcodes/Articles

This image can be printed for labeling the product or storage bin in the warehouse, so it's enabled for scanning.

With the program Sales, the sales can be established by scanning barcodes.

The amount can be set with the little spinbox. The amount can be filled with the little arrows, or turning the mousewheel on the field. The amount can be set from 1 to 99. The amount is default set to 1.

The module scans the barcode, looks up in the database the articlenumber, description, price and counts the subtotals and VAT. When scanning is completed, a orderlist can be printed.

If a item cannot be provided with a barcode, it is possible to assign the product to a button.

Press the button, after setting the amount if more than one piece required.

See explanation under heading 'Programmable buttons'.

If scanning is not possible, in case of a damaged barcode, the barcode can be filled manually.

In this case the <Enter> must be pressed on the keyboard.

The program checks if 13 numbers are filled and checks the validity check number.

The module provides a display with heading and 16 product lines.

The lines are scrolling, if more than 16 lines are added.

Below the display screen the totals including VAT and total VAT is displayed.

If the first scan is established the close button is blocked, until next client button is pressed.

Printing of the order is possible until the button next client is pressed.

In the print module the total price and total VAT is counted and printed in the tail heading.

After the next client button is pressed the print button and the next client button is blocked, until the first scan for the next client is established. By scanning the table sales is filled with the order receipt number, article number, description, amount, price, subtotal and subVAT, also in the table articles the stock data is updated.

With pressing the next client button the table afdrachten (payments) is filled with the totals and VAT.

By scanning is checked on 4 error conditions:

1. The checksum of the barcode is wrong. (wrong barcode or damaged)
2. The product is not in the range of the company.
3. Too little stock for the orderline.
4. Error message if not logged on.

The errors are showed below the product display in the color red.

With choice 3, also the amount of current stock is showed.

For return goods a ± button is added.

This button is visible if the logon barcode is valid as code 2 or code 3. See **Accounts**.

If the button (checkable) is set on – the spinbox range change from -1 to -99.

The logon can be switched from access 1 to access 2 or 3 and back with remaining of the orderlines, so it's possible to book return goods by a competent person.

### **Articles Request/change**

With this menuchoice a tableview from the articles table sorted on barcode is shown.

Change articles:

By clicking on the first field a article form is opened for changing article items.

The fields order\_size and minimum\_stock are also once a year calculated by the program, but can be overruled intermediate. Also item\_stock can be filled for new articles, or for test purposes.

Normally this should be handled by imports of delivery lists. So handle with care.

The calculation of this items is established by the Camp formula:

Camp formula:  $Q = \sqrt{2DF/HP}$

Stands for:

Q = Quantity

- D = Demand / year (table articles)
- F = Fixed Costs (order costs conversion costs) (from table params)
- H = Stock costs as a percentage of the price (from table params)
- P = Price of the product (from table articles)

The annual\_consumption is counted by the article usage in a year.

Even year annual\_consumption\_1

Odd year annual\_consumption\_2

By starting of a new year the oldest year of annual consumption is set to 0, so the counting restarts, with this column, so always the last year is preserved for calculation of stock.

It is possible to influence the minimum stock and order size by changing the paramID 5 and 6 (Ordercosts and surcharge for interest and storage)

The params 3 and 4 are changed by the program. So leave it untouched by normal operation.

### **Sales Request**

With this menuchoice a tableview from the sales sorted on receiptnumber is shown.

### **Payments Request/Paying**

With this menuchoice a tableview from the payments towards instances sorted on not payed and receiptnumber is shown.

With this menuchoice is it also possible to book a payment towards instances by clicking on the thurst field from the payments request tableview. This action opens a payform with a checkbox pay. By checking this box the payment is booked with the paydata. After this booking is done the checkbox is disabled for checking and the text from the box is changed in payed.

### **Importing.**

Imports will be common in retail for expired products, price changes and new items.

So we will produce some lists and modules for importing these items.

New items will been added, expired items will been deleted and price changes will been executed.

The import files have a fixed layout for the lines, I have chosen this, because csv files are difficult due to the many punctuation marks in the description fields of the used database.

Templates are provided in folder ./forms/Imports/Templates/

The layout for changed prices:

barcode,item\_price per line.

Filename: prices\_<dateofmakingday> without extension.

Folder ./forms/Imports/Prices/

The layout for new products:

barcode,description,item\_price,item\_unit,article\_group,thumbnail,category,VAT per line.

Filename: new\_<dateofmakingday> without extension.

Folder ./forms/Imports/New/

The layout for expired products:

barcode per line.

Filename: expired\_<dateofmakingday> without extension .

Folder ./forms/Imports/Expired/

The imports are executed by the importmenu.

The extension .txt is added after processing, so the file is blocked for processing and available for viewing.

### **Write off Loss.**

Clicking on the first field of the tableview opens a form loss.

With this form the warehouse differences, damaged products, obsolete products or shell life products can be written off. Fill in number of products and choose category of loss. The products are booked in the table loss and reduced from stock.

### **Programmable Buttons.**

Accessable with the admin button (security level 3)

There are 5 buttongroups each with 39 programmable buttons with the thirist button (colored differently) as a group button to switch between the buttongroups .

The groupbuttons can be provided or changed with your proprietary grouptext in the table params paramID 7 (button-group 1), 8 (button-group 2) , 9 (button-group 3) , 10 (button-group 4) or 11 (button\_group 5).

Buttontext until 30 positions devided over maximum 3 lines.

With the menuchoice Buttons define you will get a submenu 'New barcode' or 'Existing barcode'

The thirist choice calls a form there a new barcode is generated and fields for a new article must be filled in, also in this form the buttontext can be inserted.

The second choice displays the barcode table. By browsing to the barcodeproduct and clicking on the thirist field a form is opened for inserting buttonnumber and buttontext. By accepting the buttonnumber with it's text the button is linked to the chosen barcodenumber.

The buttons for group 1 are 1-39, the buttons for group 2 are 41-79, the buttons for group 3 are 81-119, the buttons for group 4 are 121-159 and the buttons for group 5 are 161-199.  
Leave the numbers 0, 40, 80, 120, 160 untouched, they are reserved for groupbuttons.

The buttontext is until 30 positions divided over maximum 3 lines.

If a button with a existing text is choosen the text is replaced and the button is linked towards the new barcodenumber.

Minimal the fields description, price, buttonnumber and buttontext must be completed.

The new buttontext becomes visible after a restart of the program.

### **Purchases.**

The purchases will be established by a list, which is gathered from the articles, where minimum stock is less than stock plus order balance and the item is not on order (order\_status = True)

As soon as the list is produced for ordering the order\_status of the ordered items is blocked until delivery is done. (order\_status = False)

After delivery a list is gathered from approved items. The fields on this list are barcode and delivered amounts.

This list is imported and the amounts are added to stock, the order\_balance is reduced and order status is released (order\_status = True)

The extension .txt is added after processing, so the delivery import is blocked for processing and available for viewing.

The order and delivery lists are stored for printing or reference views.

### **Parameters.**

With this menuchoice a tableview from the params table sorted on paramID is shown and with clicking on the thirst field you can change item and value.

Parameters 1 and 2 are for VAT surcharges high and low, zero is set hardcoded.

Parameters 3 and 4 are set by the system so leave it untouched, except for test purposes.

Parameter 5 is charge for ordercosts

Parameter 6 is surcharge for storage and interest costs.

Parameters 7, 8 ,9, 10 and 11 are respectively for buttongroup 1, 2, 3 ,4 and 5 buttongrouptext.

### ***Addition networks:***

As a database system is used posgreSQL, which make it possible to use this application as a client server application with several point of sales.

I have not tested this because i don't have a network available. But with the right IP configuration this should be a piece of cake with a little knowledge of networking.

The server can easily host on Linux.

***The application sales.py is ported and tested for Windows and Ubuntu (might work with other Linux operation systems too).***