Create a Custom Expense Report with GnuCash and MySQL

by

Joe Hesse

joe.hesse@actcx.com

I use gnucash to keep track of my income and expenses. I live in the US and my currency is US dollars. I recently wanted a report that gives me the total expenses for each of the 86 expense items that I have created in gnucash. The expense report that gnucash provides can only show a maximum of 24 items and “another” item for the remaining expenses.

Here is a sample of the report I want and which we will produce. Only a few lines are shown, the actual report has more rows. The “Totals” column is in descending order.

```
+------+----------+-------------------------------------------------------------+
| Year | Totals   | Expense Category                                            |
+------+----------+-------------------------------------------------------------+
| 2016 | $6055.41 | Expenses:Groceries                                          |
| 2016 | $597.37  | Expenses:Auto:Bentley:Gas                                   |
| 2016 | $823.01  | Expenses:Auto:Porsche:Gas                                   |
| 2016 | $20.55   | Expenses:Auto:Tolls&Parking                                 |
| 2016 | $20.40   | Expenses:Shipping:Domestic                                  |
+------+----------+-------------------------------------------------------------+
```

Since I have some expertise in MySQL and gnucash allows one to save data in a MySQL database, I decided to use the database to create the expense report. Gnucash on my computer is running under Fedora 27 Linux. In order to do a "Save As" to MySql in gnucash, the “mysql dbi” backend must be installed. In Fedora 27 the installation is done as:

```
# dnf install libdbi-dbd-mysql
```

I am using MariaDB as my database engine. One of the queries I am using is a recursive CTE (Common Table Expression). This feature is only implemented in the latest versions of MySQL and MariaDB. Check with the MySQL or MariaDB web site to be sure your repository has the latest version and, if not, get it from the web site.

If you are not familiar with CTE or recursive CTE there are excellent tutorials on the web. If you are using another flavor of Linux or Windows, the installation described above should be similar.

The database name that gnucash uses is “gnucash” and it has 24 tables. The tables of interest in creating a custom expense report are “accounts”, “transactions” and “splits”. As the names suggest, “accounts” has account information and “transactions” has transaction information. There is a N:N relationship between these two tables and the table joining them is “splits”.

Here is a description of the “accounts” table. The fields that we will be using in are in **bold** type.

```
MariaDB [gnucash]> describe accounts;
+---------------+--------------+------+-----+---------+-------+
| Field         | Type         | Null | Key | Default | Extra |
+---------------+--------------+------+-----+---------+-------+
| guid          | varchar(32)  | NO   | PRI | NULL    |       | <== 1
| name          | varchar(2048)| NO   |     | NULL    |       | <== 2
| account_type  | varchar(2048)| NO   |     | NULL    |       | <== 3
| commodity_guid| varchar(32)  | YES  |     | NULL    |       |
| commodity_scu | int(11)      | NO   |     | NULL    |       |
| non_std_scu   | int(11)      | NO   |     | NULL    |       |
| parent_guid   | varchar(32)  | YES  |     | NULL    |       | <== 4
| code          | varchar(2048)| YES  |     | NULL    |       |
```

```
| description    | varchar(2048) | YES  |     | NULL    |       |
| hidden         | int(11)       | YES  |     | NULL    |       |
| placeholder    | int(11)       | YES  |     | NULL    |       |
+----------------+---------------+------+-----+---------+-------+
11 rows in set (0.02 sec)
```

Item #1 is the primary key for the table. Item #3 is the type of the account. The possible values for this field are:

```
MariaDB [gnucash]> select distinct account_type from accounts order by
account_type;
+--------------+
| account_type |
+--------------+
| ASSET        |
| BANK         |
| CREDIT       |
| EQUITY       |
| EXPENSE      | <== Use this one
| INCOME       |
| LIABILITY    |
| ROOT         |
+--------------+
8 rows in set (0.00 sec)
```

Since we are interested in a custom expense report, the relevant account_type is “EXPENSE”.

Item #2 is the name of the account. It corresponds to a single “Account Name” under the gnucash “Accounts” tab. If, for example, you have an account:

`Expenses:Medical:Dentist`

Then each of the above items has its own record in the “accounts” table.

Since every account except one has a parent, Item #4 is the guid of the parent.

For the sake of completeness, we give the description of the other 2 tables. The fields that we will be using are written, as before, in **bold** type.

```
MariaDB [gnucash]> describe splits;
+-----------------+---------------+------+-----+---------------------+-------+
| Field           | Type          | Null | Key | Default             | Extra |
+-----------------+---------------+------+-----+---------------------+-------+
| guid            | varchar(32)   | NO   | PRI | NULL                |       |
| tx_guid         | varchar(32)   | NO   | MUL | NULL                |       | <== 1
| account_guid    | varchar(32)   | NO   | MUL | NULL                |       | <== 2
| memo            | varchar(2048) | NO   |     | NULL                |       |
| action          | varchar(2048) | NO   |     | NULL                |       |
| reconcile_state | varchar(1)    | NO   |     | NULL                |       |
| reconcile_date  | timestamp     | YES  |     | 0000-00-00 00:00:00 |       |
| value_num       | bigint(20)    | NO   |     | NULL                |       | <== 3
| value_denom     | bigint(20)    | NO   |     | NULL                |       |
| quantity_num    | bigint(20)    | NO   |     | NULL                |       |
| quantity_denom  | bigint(20)    | NO   |     | NULL                |       |
| lot_guid        | varchar(32)   | YES  |     | NULL                |       |
+-----------------+---------------+------+-----+---------------------+-------+
12 rows in set (0.27 sec)
```

Item #1 and Item #2 are foreign keys connecting this table to the other two.

Item #3 is the monetary amount of the transaction, in pennies.

```
MariaDB [gnucash]> describe transactions;
+---------------+---------------+------+-----+---------------------+-------+
| Field         | Type          | Null | Key | Default             | Extra |
+---------------+---------------+------+-----+---------------------+-------+
| guid          | varchar(32)   | NO   | PRI | NULL                |       | <== 1
| currency_guid | varchar(32)   | NO   |     | NULL                |       |
```

```
| num           | varchar(2048) | NO   |     | NULL                |       |
| post_date     | timestamp     | YES  | MUL | 0000-00-00 00:00:00 |       | <== 2
| enter_date    | timestamp     | YES  |     | 0000-00-00 00:00:00 |       |
| description   | varchar(2048) | YES  |     | NULL                |       |
+---------------+---------------+------+-----+---------------------+-------+
6 rows in set (0.00 sec)
```

Item #1 is the primary key for the table. Item #2 is the date of the transaction.

Before proceeding, there is one problem to be addressed. You may think that the final custom expense report will be produced by joining all 3 tables on the appropriate fields, grouping by “name” and selecting “name” and SUM on “value_num”. The problem is there may be duplicate names in the “name” column of accounts. If you proceed like this you will get the combined expenses for the duplicate name, probably not what you want. Even if there are no duplicate name, the final report is more informative if the full hierarchy of the expense item is shown. As regards duplicate names, please consider the following.

The accounts are hierarchical. For example, I could have the following 2 expense items spread over several records.:

```
Expenses:Auto:Bentley:Fuel
Expenses:Auto:Porsche:Fuel
```

The “name” field in the “accounts” table contains records for “Expenses”, “Auto”, “Bentley”, “Porsche”, “Fuel” and “Fuel”. Notice “Fuel” appears twice in 2 separate records. You can see the hierarchy by tracing through the “parent_guid” fields. If, for example, you look at the records whose “guid” is the “parent_guid” for each of these 2 records you will find “Bently” and “Porsche” as “name”. This can be repeated until you come to “Expenses”. I recommend you run the following query to see this in action for your own accounts.

```
MariaDB [gnucash]> SELECT guid, parent_guid, name FROM accounts WHERE account_type
= "Expense" ORDER BY name;
```

When a report is produced you will see the total expenses for “Fuel”.. If nothing special is done, the report will combine the “Fuel” expenses for the two autos. This is probably not what you want. You could fix the problem by having two different “Fuel” names, e.g., “FuelB” and “FuelP”. This would require you to change how you enter accounts into gnucash. You can see how many account hierarchies end in the identical name by running the following query.

```
MariaDB [gnucash]> SELECT name, COUNT(*) FROM accounts GROUP BY name HAVING
COUNT(*) > 1;
```

Fortunately, there is a better way to solve the above problem. It is all contained in the following MySQL script named “Report.sql”. If you run it as is you will get a custom expense report for the year 2016. The report can be easily modified to produce a report for any year or selection of dates and, if desired, produce an excel file.

To get the expense report on your system, put the script in a text file, say “Report.sql” and start MySQL from the folder the script is in.

```
[MySQL]$ mysql -u user -p gnucash
Enter password:

MariaDB [gnucash]> source Report.sql;
The report will print
```

Report.sql

```
USE gnucash;

# In case this script crashed earlier
DROP TABLE IF EXISTS accounts1;
DROP VIEW  IF EXISTS accounts3;

########################################################################

# Create new table accounts1 from accounts that has the fields and
# rows that we are interested in
CREATE TABLE accounts1 AS SELECT guid, name, parent_guid
FROM accounts
WHERE account_type = 'EXPENSE';

# The original VARCHAR(2048) for 'name' crashed
# the "RECURSIVE CTE" query, must decrease VARCHAR size
# VARCHAR(100) seems reasonable
ALTER TABLE accounts1 CHANGE name name VARCHAR(100);

# New accounts1 table should have a primary key
ALTER TABLE accounts1 CHANGE guid guid VARCHAR(32) PRIMARY KEY;

########################################################################

# Create view accounts3, that "looks like" accounts1 but with the
# added column 'path'
CREATE VIEW accounts3 AS
WITH RECURSIVE accounts2 AS  # RECURSIVE CTE
(
  SELECT guid, name, parent_guid, name AS path
  FROM accounts1
  WHERE name = 'Expenses'

  UNION ALL

  SELECT a1.guid, a1.name, a1.parent_guid,
         CONCAT(a2.path, ':', a1.name)
  FROM
    accounts1 AS a1 INNER JOIN accounts2 AS a2
      ON a2.guid = a1.parent_guid
)
SELECT guid, parent_guid, name, path FROM accounts2;

########################################################################

# Print report
SELECT YEAR(tr.post_date) AS Year, # <== Not necessary to print YEAR
       CONCAT('$', TRUNCATE(SUM(sp.value_num)/100.0,2)) AS Totals,
       a3.path AS 'Expense Category'
FROM
accounts3 AS a3
JOIN splits AS sp ON a3.guid=sp.account_guid
JOIN transactions AS tr ON tr.guid=sp.tx_guid
WHERE YEAR(tr.post_date) = '2016'  # <== CHANGE TO SUIT
GROUP BY a3.path
ORDER BY SUM(sp.value_num) DESC;

########################################################################

# Clean up
DROP TABLE IF EXISTS accounts1;
DROP VIEW  IF EXISTS accounts3;
```