Scanduc 25

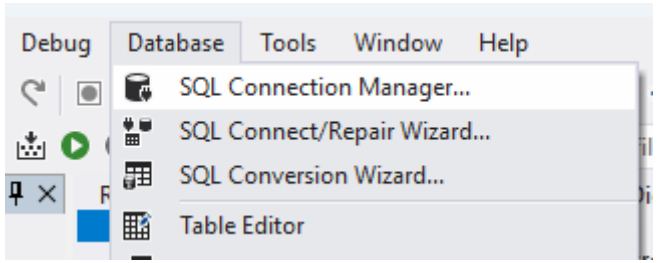
Practical SQL for DataFlex: A Rapid Overview

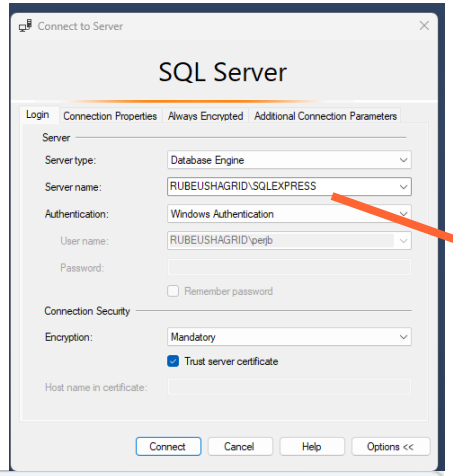
Johan Broddfelt

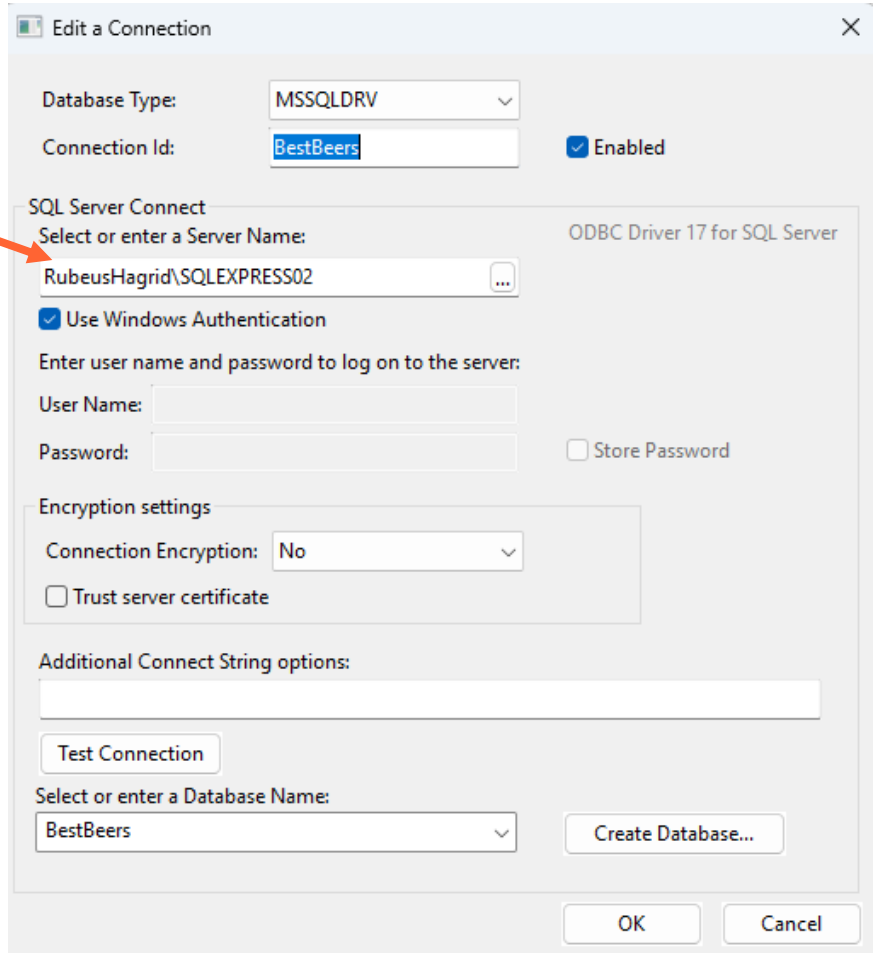
Disclaimer

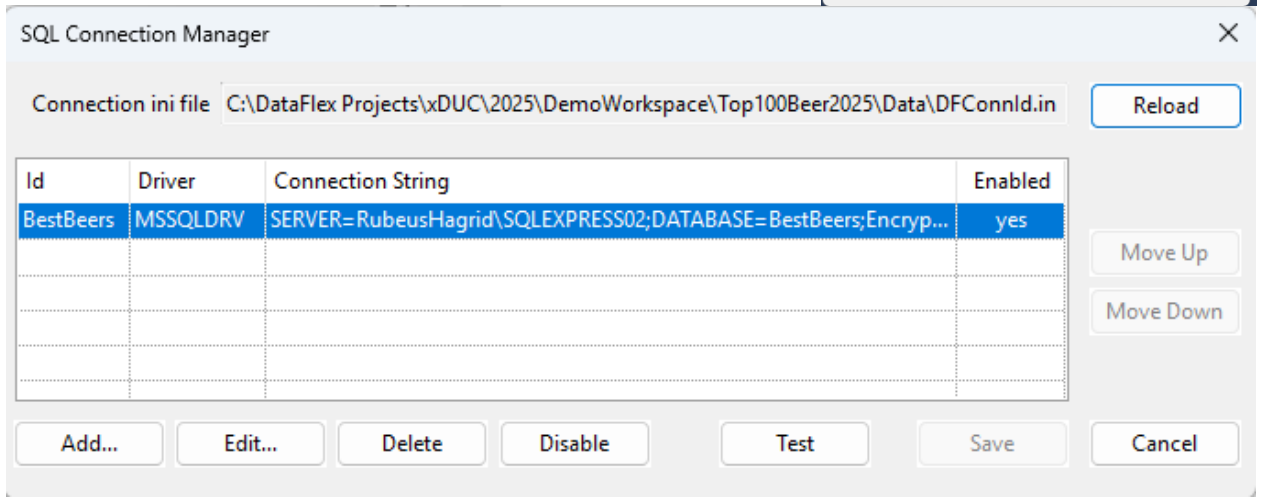
- SQL opens a can of worms to play with. Thread with care…
  - Update and read data outside DD rules
  - SQL-injection and other security concerns
- Keep the applications business rules in mind

Connect DD:s to SQL-server









Information in code

./data/DFConnId.ini

[connection1]

id=**BestBeers**

driver=MSSQLDRV

connection=SERVER=MyPC\SQLEXPRESS02;DATABASE=BestBeers;Encrypt=No;TrustServerCertificate=no

trusted_connection=yes

./data/beer.int

DRIVER_NAME MSSQLDRV

SERVER_NAME DFCONNID=**BestBeers**

DATABASE_NAME Beer

SCHEMA_NAME dbo

TopBeersFlx.src

```
Object oApplication is a cApplication
    Set peHelpType to htHtmlHelp

    Object oConnection is a cConnection
        Use LoginEncryption.pkg
        Use DatabaseLoginDialog.dg
    End_Object

End_Object
```

Connect other SQL-queries to DB

MainSqlConnection.pkg

```
// Defines an object of cSQLExecutor class
// Can be be used as a singleton object in a program
// Use global variable ghoSQLExecutor to access the SQLExecutor instance
Use cSQLExecutor.pkg
//Use SqlEx\cSQLExecutorErrorLogging.pkg

Object oMainSqlConnection is a cSQLExecutor //ErrorLogging
    Move Self to ghoSQLExecutor

    Set psConnectionId to "BestBeers"

    Procedure OnSQLError String sSQLState String sSQLMessage
        Forward Send OnSQLError sSQLState sSQLMessage
    End_Procedure
End_Object
```

TopBeersFlx.src

```
Use MainSqlConnection.pkg
```

Old method

```
Object oSQLHandler is a cSQLHandleManager
    // From DFAllEnt.pkg > cConnection.pkg > sql.pkg
    Move Self to hoSQLMngr
End_Object
```

Identity column

DataDictionary configuration

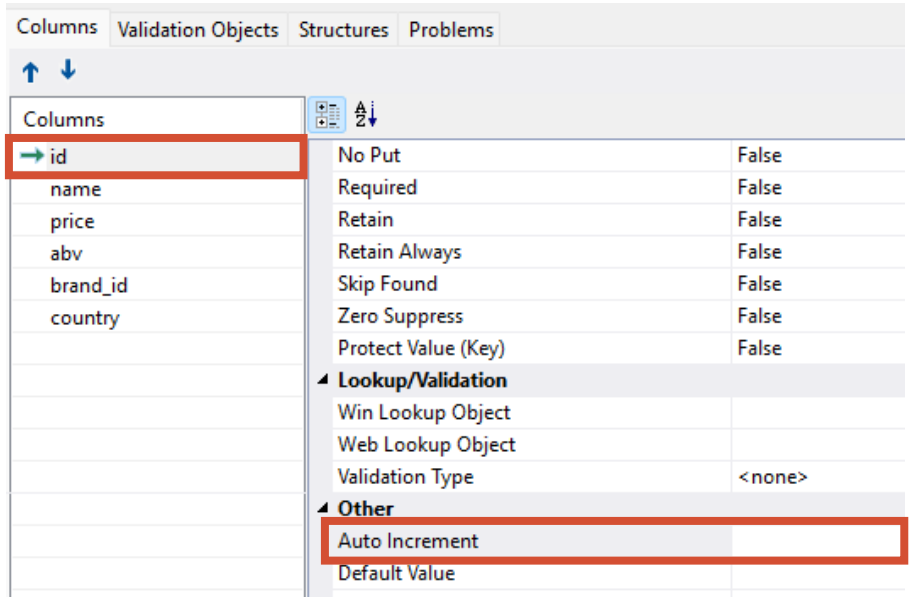
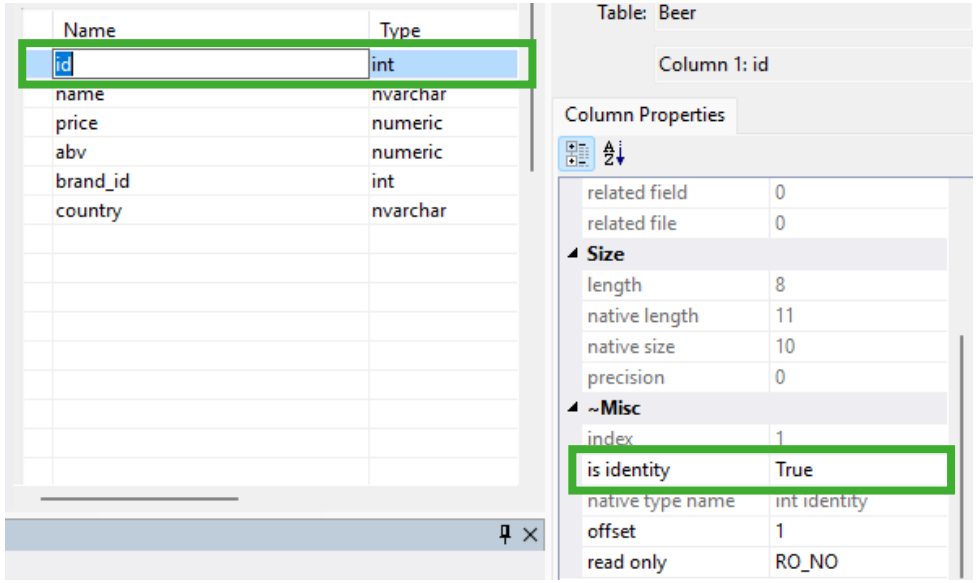


Table configuration



Old vs New SQL-commands in DataFlex

Connect other SQL-queries to DB

Using the Record Buffer

```
Clear RouteStore
Move iStoreId to RouteStore.StoreId
Move iRouteId to RouteStore.RouteId
Find EQ RouteStore by Index.1
If (Found) Begin
    Move dtNow to RouteStore.PickupReportSent
    SaveRecord RouteStore
End
```

New vs Old SQL-methods

Using new SQL functions

```
Include_Text SQL/UpdateRouteStoreExported.sql as C_RSExported
Send SQLPrepare of ghoSQLExecutor C_RSExported
Send SQLSetParameter of ghoSQLExecutor "storeId" iStoreId
Send SQLSetParameter of ghoSQLExecutor "routeId" iRouteId
Send SQLExecute of ghoSQLExecutor
```

UpdateRouteStoreExported.sql

```
UPDATE [dbo].[RouteStore]
SET PickupReportSent = CURRENT_TIMESTAMP
WHERE storeId = ${storeId}
AND routeId = ${routeId}
```

Using old SQL functions

```
Handle hoSQLMngr hdbc hStmt
Object oSQLHandler is a cSQLHandleManager
   Move Self to hoSQLMngr
End_Object

Get SQLFileConnect of hoSQLMngr RouteStore.File_number to hdbc
Get SQLOpen of hdbc to hStmt

Move @"UPDATE [dbo].[RouteStore]
        SET PickupReportSent = CURRENT_TIMESTAMP
       WHERE storeId = " + iStoreId + "
        AND routeId = " + iRouteId to sQuery
Send SQLExecDirect of hStmt sQuery

Send SQLClose of hstmt
Send SQLDisconnect of hdbc


// The following methods in the old that is not working in the new version
SQLFetch and SQLColumnValue
```

Include SQL-file

Using new SQL functions

```
Include_Text SQL/UpdateRouteStoreExported.sql as C_RSExported
Send SQLPrepare of ghoSQLExecutor C_RSExported
Send SQLSetParameter of ghoSQLExecutor "storeId" iStoreId
Send SQLSetParameter of ghoSQLExecutor "routeId" iRouteId
Send SQLExecute of ghoSQLExecutor
```

UpdateRouteStoreExported.sql

```
UPDATE [dbo].[RouteStore]
SET PickupReportSent = CURRENT_TIMESTAMP
WHERE storeId = ${storeId}
AND routeId = ${routeId}
```

Using old SQL functions

```
Handle hoSQLMngr hdbc hStmt
Object oSQLHandler is a cSQLHandleManager
   Move Self to hoSQLMngr
End_Object

Get SQLFileConnect of hoSQLMngr RouteStore.File_number to hdbc
Get SQLOpen of hdbc to hStmt

Move @"UPDATE [dbo].[RouteStore]
        SET PickupReportSent = CURRENT_TIMESTAMP
       WHERE storeId = " + iStoreId + "
        AND routeId = " + iRouteId to sQuery
Send SQLExecDirect of hStmt sQuery

Send SQLClose of hstmt
Send SQLDisconnect of hdbc


// The following methods in the old that is not working in the new version
SQLFetch and SQLColumnValue
```

Risk for SQL-injection

Using new SQL functions

Include_Text S(
Send SQLPrepa
Send SQLSetPa
Send SQLSetPa
Send **SQLExecu**

UpdateRo
UPDATE [dbo].
SET PickupRepo
WHERE storeId
AND routeId =

Using old SQL functions

```
Get SQLEscapedStr iStoreId to iStoreId                  // Changes ' to ''
Get SQLEscapeLikeWildcards iStoreId to iStoreId     // Changes % to \%
Get SQLEscapedStr iRouteId to iRouteId                  // Changes ' to ''
Get SQLEscapeLikeWildcards iRouteId to iRouteId     // Changes % to \%

Move @"UPDATE [dbo].[RouteStore]
        SET PickupReportSent = CURRENT_TIMESTAMP
      WHERE storeId = '' + iStoreId + ''
        AND routeId = '' + iRouteId to sQuery

// This does both SQLEscapedStr and SQLEscapeLikeWildcards
Get SQLStrLike (RefTable(RouteStore.storeId)) iStoreId to sFilter
```

hdbc

// The following methods in the old that is not working in the new version
SQLFetch and **SQLColumnValue**

Less code to maintain

Using new SQL functions

```
Include_Text SQL/UpdateRouteStoreExported.sql as C_RSExported
Send SQLPrepare of ghoSQLExecutor C_RSExported
Send SQLSetParameter of ghoSQLExecutor "storeId" iStoreId
Send SQLSetParameter of ghoSQLExecutor "routeId" iRouteId
Send SQLExecute of ghoSQLExecutor
```

UpdateRouteStoreExported.sql

```
UPDATE [dbo].[RouteStore]
SET PickupReportSent = CURRENT_TIMESTAMP
WHERE storeId = ${storeId}
AND routeId = ${routeId}
```

Using old SQL functions

```
Handle hoSQLMngr hdbc hStmt
Object oSQLHandler is a cSQLHandleManager
   Move Self to hoSQLMngr
End_Object

Get SQLFileConnect of hoSQLMngr RouteStore.File_number to hdbc
Get SQLOpen of hdbc to hStmt

Move @"UPDATE [dbo].[RouteStore]
        SET PickupReportSent = CURRENT_TIMESTAMP
       WHERE storeId = " + iStoreId + "
        AND routeId = " + iRouteId to sQuery
Send SQLExecDirect of hStmt sQuery

Send SQLClose of hstmt
Send SQLDisconnect of hdbc


// The following methods in the old that is not working in the new version
SQLFetch and SQLColumnValue
```

Constant name need to be unique

Using new SQL functions

Include_Text SQL/UpdateRouteStoreExported.sql as C_RSExported
Send **SQLPrepare** of ghoSQLExecutor C_RSExported
Send **SQLSetParameter** of ghoSQLExecutor "storeId" iStoreId
Send **SQLSetParameter** of ghoSQLExecutor "routeId" iRouteId
Send **SQLExecute** of ghoSQLExecutor

UpdateRouteStoreExported.sql

UPDATE [dbo].[RouteStore]
SET PickupReportSent = CURRENT_TIMESTAMP
WHERE storeId = ${storeId}
AND routeId = ${routeId}

```
Include_Text UpdateCustomers.sql as C_SQLUpdateCustomers

Variant vResult

Get SqlExecDirect of ghoSQLExecutor C_SQLUpdateCustomers to vResult
```

Move C_RSExported to **sQuery**
Send **SQLPrepare** of ghoSQLExecutor **sQuery**

Using old SQL functions

```
Handle hoSQLMngr hdbc hStmt
Object oSQLHandler is a cSQLHandleManager
   Move Self to hoSQLMngr
End_Object

Get SQLFileConnect of hoSQLMngr RouteStore.File_number to hdbc
Get SQLOpen of hdbc to hStmt

Move @"UPDATE [dbo].[RouteStore]
        SET PickupReportSent = CURRENT_TIMESTAMP
       WHERE storeId = " + iStoreId + "
        AND routeId = " + iRouteId to sQuery
Send SQLExecDirect of hStmt sQuery

Send SQLClose of hstmt
Send SQLDisconnect of hdbc


// The following methods in the old that is not working in the new version
SQLFetch and SQLColumnValue
```

Large queries *(obsolete)*

```
SELECT *
FROM (SELECT bc AS pallet_ref,
             IIF(SO.closed=1,'Delivered','In transit') AS del_status,
             SO.label,
             SO.detail,
             ST.name,
             SP.name AS shipr,
             O.collection_date,
             O.original_eta,
             IIF(O.original_eta=O.eta,'',CONVERT(VARCHAR(10),O.eta,23)) AS eta,
             IIF(O.delivery_date<>'0001-01-01' AND O.delivery_time<>'',CONCAT(O.delivery_date,' ',O.delivery_time),'') AS DelTime,
             ST.country,
             PT.parner_number,
             MONTH(O.eta) AS mm,
             DAY(O.eta) AS dd,
             YEAR(O.eta) AS yy,
             ch_rsn AS change_reason,
             ocr_rsn AS other_change_reason,
             c_rsn AS comment,
             ST.delivery_from AS sdf,
             ST.delivery_to AS sdt,
             O.delivery_from,
             O.delivery_to,
             O.reference,
             O.pallet_count,
             IIF(O.delivery_from=ST.delivery_from AND O.delivery_to=ST.delivery_to,'0','1') AS ChSOP,
             ST.porterage,
             ST.fix_delivery,
             ST.out_of_hours,
             IIF(O.on_time=1,'On Time','Late') AS OnTime,
             IIF(O.on_time=1,'',
                 IIF(O.respid=1 AND O.pstatid>2,'Customer delay',
                 IIF(O.respid=0 AND O.pstatid>2,'Carrier delay',
                 IIF(O.pstatid<3,'Pending',
                 IIF(O.respid=2 AND O.pstatid>2,'Other',
                 IIF(O.respid=3 AND O.pstatid>2,'Both',
                 '')))))) AS Responsible,
             KA.name AS kpia,
             actually_collected_pallets,
             PR.reason,
             pallets_manually_corrected,
             O.created,
             O.pstatid AS status_id,
             PA.partner,
             O.delivery_date,
             KA.id AS kpid
      FROM (SELECT id AS partner FROM [BWSPortal].[dbo].[Partner]) PA
      LEFT JOIN [BWSPortal].[dbo].[Store] ST
      ON ST.partner_id=PA.partner
      LEFT JOIN [BWSPortal].[dbo].[KPIArea] KA
      ON ST.kpiarea_id=KA.id
      LEFT JOIN (SELECT [id],[reference],[created],shipper_id,[store_id],pallet_count,[collection_date],[eta]
                        ,IIF(OO.original_eta=OO.delivery_date AND ((OO.delivery_from < OO.delivery_to) AND (OO.delivery_time BETWEEN OO.delivery_from AND OO.delivery_to) OR
                        (OO.delivery_from > OO.delivery_to) AND NOT(OO.delivery_time BETWEEN OO.delivery_from AND OO.delivery_to))),1,0) AS on_time
                        ,[delivery_from],[delivery_to]
                        ,[original_eta],[created_in_system],[delivery_date],[delivery_time],[org_collection_date]
                        ,[actually_collected_pallets],[pallet_reason_id],[pallets_manually_corrected]
                        ,ch_rsn,ocr_rsn,c_rsn,bc
                        ,pstatid,respid
                 FROM (SELECT SH.[id],SH.[reference],SH.[created],SH.shipper_id,SH.[store_id],SH.pallet_count,SH.[collection_date],SH.[eta]
                              ,DATEADD(MINUTE,-STO.delivery_window_start,CONVERT(TIME,IIF(SH.[delivery_from]<>'',SH.[delivery_from],'0G:00'))) AS delivery_from
                              ,DATEADD(MINUTE,STO.delivery_window_end,CONVERT(TIME,IIF(SH.[delivery_to]<>'',SH.[delivery_to],'18:00'))) AS delivery_to
                              ,SH.[original_eta],SH.[created_in_system],SH.[delivery_date],SH.[delivery_time],SH.[org_collection_date],STRING_AGG(P.barcode,',') AS bc
                              ,STRING_AGG(IIF(TRIM(P.change_reason)<>'',TRIM(P.change_reason),NULL),',') AS ch_rsn
                              ,STRING_AGG(IIF(TRIM(P.other_change_reason)<>'',TRIM(P.other_change_reason),NULL),',') AS ocr_rsn,STRING_AGG(P.comment,',') AS c_rsn
                              ,MAX(P.status_id) AS pstatid,MAX(HR.responsible) AS respid
                              ,[actually_collected_pallets],[pallet_reason_id],[pallets_manually_corrected]
                       FROM [BWSPortal].[dbo].[Shipment] SH
                       LEFT JOIN [BWSPortal].[dbo].[pallet] P
                       ON P.shipment_id=SH.id
                       LEFT JOIN [BWSPortal].[dbo].[status] SS
                       ON SS.id=P.status_id
                       LEFT JOIN [BWSPortal].[dbo].[OnHoldReasons] HR
                       ON HR.reason=P.change_reason
                       LEFT JOIN (SELECT id, IIF(delivery_window_start='', 15, delivery_window_start) AS delivery_window_start, IIF(delivery_window_end='', 15, delivery_window_end) AS delivery_window_end FROM [BWSPortal].[dbo].[Store]) STO
                       ON SH.store_id=STO.id
                       GROUP BY SH.id,SH.reference,[created],shipper_id,[store_id],pallet_count,[collection_date],[eta],[delivery_from],[delivery_to],[original_eta],
                                [created_in_system],[delivery_date],[delivery_time],[org_collection_date]
                                ,[actually_collected_pallets],[pallet_reason_id],[pallets_manually_corrected]
                                ,STO.delivery_window_start,STO.delivery_window_end
                 ) OO
      ) O ON O.store_id=ST.id
      LEFT JOIN [BWSPortal].[dbo].[status] SO ON SO.id=O.pstatid
      LEFT JOIN [BWSPortal].[dbo].[shipper] SP
      ON SP.id=O.shipper_id
      LEFT JOIN [BWSPortal].[dbo].[partner] PT
      ON PT.id=PA.partner
      LEFT JOIN [BWSPortal].[dbo].[PalletReasons] PR
      ON PR.id=O.pallet_reason_id
       __sStoreFilter__
) DT
__sMainFilter__
```

```
Move 'PENDING DATA' to sPending
Move 'MAIN FILTER' to sMainFilter
Move 'STORE FILTER' to sStoreFilter

Include_Text SQL/Test.sql as C_TestQry
Move (Replaces('__sPending__', C_TestQry, sPending)) to C_TestQry
Move (Replace('__sMainFilter__', C_TestQry, sMainFilter)) to C_TestQry
Move (Replace('__sStoreFilter__', C_TestQry, sStoreFilter)) to C_TestQry
```

Fixing my mistake

IncludeSqlQueries.pkg*

```
01 Include_Text SQL\InsertBeerCategory.sql as C_InserBeerCategory
02 Include_Text SQL\InsertBeer.sql as C_InsertBeer
03 Include_Text SQL\InsertBrand.sql as C_InsertBrand
04 Include_Text SQL\InsertCategory.sql as C_InsertCategory
05 Include_Text SQL\SelectBeerByCountry.sql as C_SelectBeerByCountry
06
07 Include_Text SQL\SelectBeerQuery.sql as C_SelectBeerQuery
08 Include_Text SQL\SelectBeerQuery.sql as C_SelectBeerQueryD
09
10 Include_Text SQL\SelectBeer.sql as C_SelectBeer
11 Include_Text SQL\SelectBrand.sql as C_SelectBrand
12
13 Include_Text SQL\SelectCategoryByBeer.sql as C_SelectCategoryByBeer
14 Include_Text SQL\SelectCategoryByBeer.sql as C_SelectCategoryByBeerQ
15
16 Include_Text SQL\SelectCategory.sql as C_SelectCategory
17
18 Include_Text SQL\StarSelectCategory.sql as C_StarSelectCategory
19 Include_Text SQL\StarSelectCategory.sql as C_StarSelectCategoryQ
20
21 Include_Text SQL\StarSelectCountry.sql as C_StarSelectCountry
22 Include_Text SQL\StarSelectCountry.sql as C_StarSelectCountryQ
23
24 Include_Text SQL\TruncateAll.sql as C_TruncateAll
25
```

Better structure

IncludeSqlQueries.pkg*

```
01 Include_Text SQL\InsertBeerCategory.sql as C_InserBeerCategory
02 Include_Text SQL\InsertBeer.sql as C_InsertBeer
03 Include_Text SQL\InsertBrand.sql as C_InsertBrand
04 Include_Text SQL\InsertCategory.sql as C_InsertCategory
05 Include_Text SQL\SelectBeerByCountry.sql as C_SelectBeerByCountry
06 Include_Text SQL\SelectBeerQuery.sql as C_SelectBeerQuery
07 Include_Text SQL\SelectBeer.sql as C_SelectBeer
08 Include_Text SQL\SelectBrand.sql as C_SelectBrand
09 Include_Text SQL\SelectCategoryByBeer.sql as C_SelectCategoryByBeer
10 Include_Text SQL\SelectCategory.sql as C_SelectCategory
11 Include_Text SQL\StarSelectCategory.sql as C_StarSelectCategory
12 Include_Text SQL\StarSelectCountry.sql as C_StarSelectCountry
13 Include_Text SQL\TruncateAll.sql as C_TruncateAll
```

TopBeersFlx.src

```
Use SQL\IncludeSqlQueries.pkg
Use MainSqlConnection.pkg
```

Top100Beer2025 › AppSrc › SQL

Namn	Senast ändrad	Typ	Storlek
IncludeSqlQueries.pkg	2025-04-04 08:39	PKG-fil	1 kB
InsertBeer.sql	2025-03-05 14:16	SQL Source File	1 kB
InsertBeerCategory.sql	2025-03-27 08:32	SQL Source File	1 kB
InsertBrand.sql	2025-03-27 08:32	SQL Source File	1 kB
InsertCategory.sql	2025-03-05 14:01	SQL Source File	1 kB
SelectBeer.sql	2025-03-07 00:30	SQL Source File	1 kB
SelectBeerByCountry.sql	2025-03-27 08:32	SQL Source File	1 kB
SelectBeerQuery.sql	2025-03-27 08:32	SQL Source File	1 kB
SelectBrand.sql	2025-03-27 08:32	SQL Source File	1 kB
SelectCategory.sql	2025-03-05 14:13	SQL Source File	1 kB
SelectCategoryByBeer.sql	2025-03-06 22:01	SQL Source File	1 kB
StarSelectCategory.sql	2025-03-27 08:32	SQL Source File	1 kB
StarSelectCountry.sql	2025-03-06 22:48	SQL Source File	1 kB
Test.sql	2025-04-03 10:07	SQL Source File	6 kB
TruncateAll.sql	2025-03-06 16:12	SQL Source File	1 kB

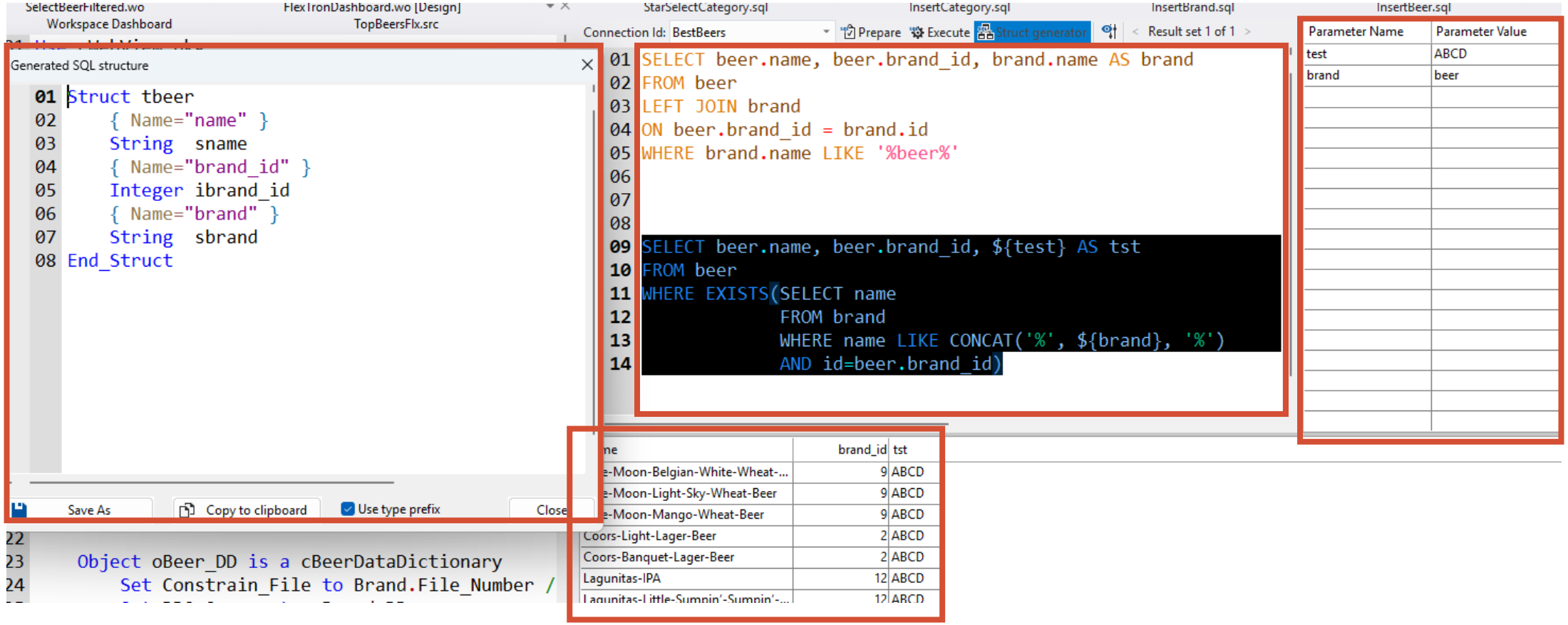
WARNING!!!

```
Move 'PENDING DATA' to sPending
Move 'MAIN FILTER' to sMainFilter
Move 'STORE FILTER' to sStoreFilter

Include_Text SQL/Test.sql as C_TestQry
Move (Replaces('__sPending__', C_TestQry, sPending)) to C_TestQry
Move (Replace('__sMainFilter__', C_TestQry, sMainFilter)) to C_TestQry
Move (Replace('__sStoreFilter__', C_TestQry, sStoreFilter)) to C_TestQry
```

Move C_RSExported to sQuery
Move (Replace('__sStoreFilter__', sQuery, sStoreFilter)) to sQuery
Send **SQLPrepare** of ghoSQLExecutor sQuery

Using the query builder



Get data into result with ARRAY

```
String[] aResult

Include_Text SQL/SelectBeer.sql as C_RSExported
Send SQLPrepare of ghoSQLExecutor C_RSExported
Get SQLExecute of ghoSQLExecutor to aResult

Move 0 to iRow
While (iRow < SizeOfArray(aResult))
   Move aResult[iRow].[0] to aSuggestions[iRow].sRowId
   Move aResult[iRow].[1] to aSuggestions[iRow].aValues[0]
   Increment iRow
Loop
```

Get data into result with STRUCT

```
tBeer[] aResult

Include_Text SQL/SelectBeer.sql as C_RSExported
Send SQLPrepare of ghoSQLExecutor C_RSExported
Get SQLExecute of ghoSQLExecutor to aResult

Move 0 to iRow
While (iRow < SizeOfArray(aResult))
   Move aResult[iRow].iid to aSuggestions[iRow].sRowId
   Move aResult[iRow].sbeer to aSuggestions[iRow].aValues[0]
   Increment iRow
Loop
```

```
Struct tBeer
   { Name="id" }
   Integer iid
   { Name="beer" }
   String  sbeer
End_Struct
```

Basic queries

```
INSERT INTO [table]
([column1], [column2])
VALUES (value1a, value2a) , (value1b, value2b) , (value1c, value2c)…
```

```
UPDATE [table]
SET [column1]=value1,
    [column2]=value2
WHERE column3=value3
```

```
DELETE [table]
WHERE column3=value3
```

```
SELECT [column1], [column2]
FROM [table]
WHERE column3=value3
```

Samples of queries

SQL/InsertBrand.sql

```
IF NOT EXISTS (SELECT 1 FROM [Brand] WHERE [name] = ${name})
BEGIN
    INSERT INTO [Brand] ([name]) VALUES (${name});
END;
```

SQL/TruncateAll.sql

```
TRUNCATE TABLE Beer;
TRUNCATE TABLE Brand;
TRUNCATE TABLE Beer_Category;
TRUNCATE TABLE Category;
```

SQL/InsertBeerCategory.sql

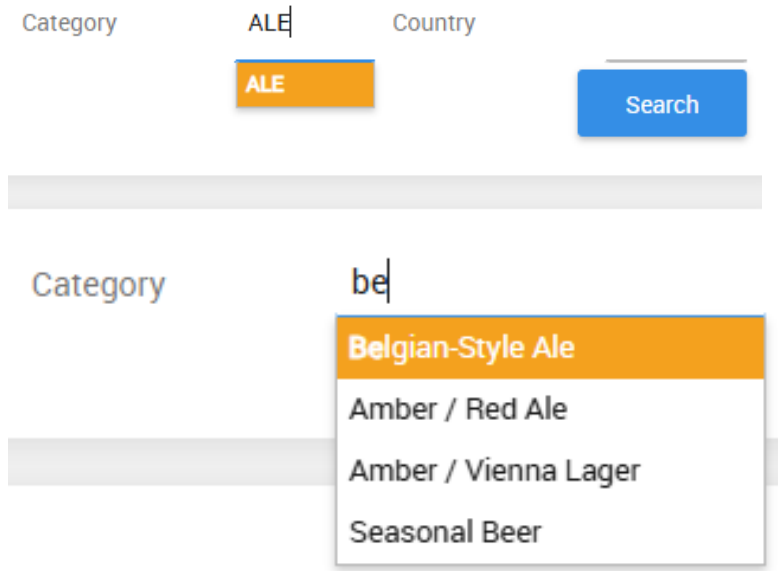
```
IF NOT EXISTS (SELECT 1
                 FROM [beer_category]
                WHERE [beer_id] = ${beer_id}
                  AND [category_id]=${category_id}
              )
BEGIN
    INSERT INTO [beer_category] ([beer_id], [category_id])
        VALUES (${beer_id}, ${category_id});
END;
```

SQL/SelectBrand.sql

```
SELECT [id]
FROM [Brand]
WHERE [name] = ${name}
```

Sorting and filters

```
SELECT [id], [name],
    CASE
        WHEN [name] LIKE ${name} THEN 2
        WHEN [name] LIKE CONCAT(${name}, '%') THEN 1
        ELSE 0
    END AS score
FROM [Category]
WHERE [name] LIKE CONCAT('%', ${name}, '%')
   OR ${name} = ''
ORDER BY score DESC, [name]
```




```
IIF([Beer].[price]<5, 'CHEAP', 'PRICY') AS cost

IIF([Beer].[price]<5, 'CHEAP',
IIF([Beer].[price]<5, 'FAIR',
    'PRICY')) AS cost
```

Aggregate queries

```
SELECT DISTINCT [Beer].[id]
      ,[Brand].[name] AS brand
      ,[Beer].[name] AS beer
      ,[Beer].[price]
      ,[Beer].[abv]
      ,[Beer].[brand_id]
      ,cat.category AS category
      ,[Beer].[country]
      ,([price]/[abv]) AS ppa
FROM [Beer]
LEFT JOIN [Brand]
ON [Brand].[id]=[Beer].[brand_id]
LEFT JOIN (SELECT beer_category.beer_id,
                  STRING_AGG(name, ', ') AS category
           FROM category
           LEFT JOIN beer_category
           ON [category].[id]=[beer_category].[category_id]
           GROUP BY beer_category.beer_id) cat
ON [beer].[id]=[cat].[beer_id]
LEFT JOIN beer_category
ON [beer].[id]=[beer_category].[beer_id]
```

```
ORDER BY brand
ORDER BY [Brand].[id]
ORDER BY ppa
```

Constraints SQL-injection risk

Avoid writing you own SQL-queries in filters.
Always use the **SQLEscapedStr** and **SQLEscapeLikeWildcards** if you need to write some custom query that introduces SQL-injection possibilities.

```
Get SQLStrLike (RefTable(beer.name)) sf.sText to sFilter // Does all below for you
// Get SQLEscapedStr sf.sText to sf.sText                // Changes ' to ''
// Get SQLEscapeLikeWildcards sf.sText to sf.sText       // Changes % to \%
// Move (" [name] LIKE '%" + sf.sText + "%' ") to sFilter // Only running this opens for SQL-injection
```

"name" LIKE N'%**Ams**%'

"name" LIKE N'%Ams%' OR **"price" < 5** OR "name" LIKE N'%Ams%’

You could try just typing **'** in the form to see if it fails in order to get an indication if it can be hacked.
_ and **%** area also special characters where **_** represents any character. B_ll will return values for both Ball and Bell and **%** is wildcard in the beginning or ending of the string.

EXISTS instead of JOIN in Constraints

```
SELECT beer.name, beer.brand_id, brand.name AS brand
FROM beer
LEFT JOIN brand
ON beer.brand_id = brand.id
WHERE brand.name LIKE '%beer%'
```

```
SELECT beer.name, beer.brand_id
FROM beer
WHERE EXISTS(SELECT name
        FROM brand
        WHERE name LIKE '%beer%'
        AND id = beer.brand_id)
```

Sorting data in Constraint

```
Set peDbGridType to gtAllData // Is needed to allow DF to sort in the view and not based on indexes
```

Then we sort by using **WebSet piSortColumn of oList to COLUMN_IDX**

```
If (sf.sSorting = 'az_brand') WebSet piSortColumn of oList to 0
If (sf.sSorting = 'az_beer') WebSet piSortColumn of oList to 1
If (sf.sSorting = 'az_price') WebSet piSortColumn of oList to 2
If (sf.sSorting = 'az_abv') WebSet piSortColumn of oList to 3
If (sf.sSorting = 'ppa') WebSet piSortColumn of oList to 6
```

Compare with ORDER BY used in regular SQL-query

```
If (sf.sSorting = 'az_brand') Move (sFilter + ' ORDER BY brand ') to sFilter
If (sf.sSorting = 'az_beer') Move (sFilter + ' ORDER BY beer ') to sFilter
If (sf.sSorting = 'az_price') Move (sFilter + ' ORDER BY price ') to sFilter
If (sf.sSorting = 'az_abv') Move (sFilter + ' ORDER BY abv ') to sFilter
If (sf.sSorting = 'ppa') Move (sFilter + ' ORDER BY ppa ') to sFilter
```

Methods in SQL

```
SELECT [country] AS [category],
       COUNT(*) AS [number]
FROM [Beer]
GROUP BY [country]
ORDER BY [number]
```

Aggregators used with GROUP BY
MAX, MIN, AVG, ABS, COUNT, STRING_AGG *(GROUP_CONCAT in MySQL)*

Functions
CONCAT, LEFT, RIGHT, SUBSTRING, GETDATE, LEN

Manipulation
CAST, CONVERT, FORMAT, REPLACE, UNICODE, UPPER, LOWER, ROUND, FLOOR

Comparing
DATEDIFF, DIFFERENCE, SOUNDEX, IIF, CASE/WHEN/THEN/END

Structure
(INNER, LEFT, RIGHT, OUTER) JOIN ON, EXISTS, UNION

Resources

Resources
https://learn.microsoft.com/en-us/sql/sql-server/educational-sql-resources
https://dev.mysql.com/doc/refman/9.0/en/

AI is also really good at creating queries and suggesting solutions

Scanduc|25

Thank you!

Are there any questions?