Data fra eksterne kilder – placering, metode og teknisk dokumentation

Placering

Data fra eksterne kilder er placeret som tabeller i en relationel database på samme fysiske databaseserver som Økonomidatabasen for at give en enkel adgang for brugere af Økonomidatabasen til at integrere de eksterne data i analyser af økonomidata overført direkte fra Ø90.

Databaseserveren er en Microsoft SQL 2016 server således at alle de indhøstede data er tilgængelige fra analyseværktøjer med standard Sql-forespørgsler, f.eks. Microsoft Excel, Power BI og R.

Databasen ”KundeAnalyseDB” er det sted hvor data-tabeller der kommer fra såvel automatiske som manuelle opdateringer er placeret.

Dokumentation af tilgængelige data.

En oversigt over de tilgængelige data (tabeller) i KundeAnalyseDB samt beskrivelser af tabellernes felter ajourføres i dokumentationssystemet ”Confluence” (https://confluence.seges.dk/display/KT/KundeAnalyseDB+-+eksterne+data).

ETL – Extract, Transform, Load.

For at oprette og opdatere data i tabellerne udføres en række operationer i forhold til at hente, formatere/strukturere og indlæse data i Sql format, den såkaldte ETL process.

Da mange data hentes manuelt og de forskellige kilder kan levere data i mange forskellige formater benyttes der flere forskellige metoder til opdateringen.

Det er væsentligt at medarbejdere i såvel Ø&V og IT kontinuerligt er opmærksomme på at data der modtages udefra og som har relevans for økonomiske analyser løbende placeres i det fælles datalager og dokumenteres i Confluence.

Extract, Transform

Der er to forskellige modeller for at fremskaffe data: ”pull”, hvor data hentes fra leverandøren feks som download, og ”push”, hvor vi modtager en dataleverance fra en eksterne leverandør.

Windows script-sproget Powershell er velegnet til at hente data fra feks web-tjenester som mange offentlige data ligger i. Alternativt kan også R være en mulighed for at hente data. Begge disse metoder kan også bruges til upload af data til databaseserveren.

Data vil hyppigt komme i kommasepareret format (csv), eller kunne konverteres til og gemmes i dette format. Der kan også være tale om at data her beriges eller ”klippes op” så der fjernes eller tilføjes kolonner. Ligeledes findes der forskellige kodningsstandarder for danske tegn som der skal tages højde for.

Når data modtages fra eksterne leverandører er det hensigtsmæssigt at undgå Excel som format, idet det besværliggør enkel og automatiseret upload. Som hovedregel vil et Excel-ark manuelt skulle gemmes i kommasepareret format før det kan upload’es. Læsning af Excel i såvel R som Powershell er muligt, men

besværligt og rammes let af formatteringsfejl, ligesom Excel's ustrukturerede datastruktur ofte modarbejder den strukturerede tabulære tilgang i SQL.

Det kan i nogle tilfælde være nødvendigt med en mere omfattende konvertering af data der modtages udefra, feks når data skal udledes fra tekst-stringe i datagrundlaget, og en sådan konvertering kan nødvendiggøre specialprogrammering enten før eller efter data overføres til SQL.

Load

Ulempen ved kommaseparerede (csv) filer er at de ikke har en definition af kolonnernes indhold, dvs om det er dato, tal, tekst osv. Det er informationer som SQL skal bruge for at lagre data korrekt. Til at konvertere data fra csv-formatet til SQL-tabeller er der udviklet et værktøj, "CsvToSqlServer", der kan foretage en analyse af en csv fil for at fastslå kolonnernes formattering, samt danne en SQL-tabel og overføre data til denne tabel, enten ved at erstatte tabellen eller indsætte rækker. "CsvToSqlServer" ernærmere beskrevet nedenfor.

Til overførsel af data der kommer fra GIS-systemet MapInfo anvendes standard-værktøjet "EasyLoader" som leveres af producenten. Det er ligeledes beskrevet nedenfor.

Manuel ETL

Praksis har vist at en lang række data opstår og modtages på en lang række måder og ad kanaler der ikke lader sig automatisere. Det være sig data fra Excel eller data der udtrækkes manuelt fra feks Ø90.

Disse data er typisk af "pull"-typen. Typisk vil de dog kunne gøres semiautomatiske ved at bygge faste scripts med brug af de tidligere nævnte værktøjer til at overføre disse data når de modtages.

Automatisk ETL

Data af "pull" typen kan automatisk hentes i faste kørsler, feks månedlige kørsler.

For at danne en automatisk datahentning laves et Powershell-script som sørger for hentning, formattering og upload. Der dannes ét script for hvert datasæt og dette idriftsættes på SQL serveren. Der er udviklet et program, "OedbEksterneData" der fungerer som en ramme omkring afviklingen af et eller flere Powershell script og som sørger for at logge jobforløbet samt sende kvitteringsmail til en administrator.

Data af "push" typen er der pt ikke etableret muligheder for, ud over at et eksternt system periodevist vil kunne sende data via mail til en medarnejder hos SEGES.

Brug af Sql data

For at data skal kunne "kædes sammen" med Økonomidatabasen (JOIN'es i SQL sprog) er det nødvendigt at de rummer en identifikation, feks CVR-nummer eller Kreds/Ejendomsnummer. I databasen OEDB findes der funktioner og virtuelle tabeller ("Views") der forenkler denne proces.

Forespørgsler vha standard SQL kan ske i værktøjet "Sql Server Management Studio" (https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms).

perh@seges.dk

Fra Excel (2016) og Power BI for Desktop (https://powerbi.microsoft.com/en-us/downloads/) kan der direkte kobles op mod Sql-server via fanen ”Data” og det indbyggede PowerQuery værktøj.

Fra R benyttes pakken RODBC til at forbinde mod Sql Server. Login informationer må aldrig gemmes i script og scriptet her viser en metode til at gemme sin personlige login, samt hvorledes data kan hentes:

```
if (!require("RODBC")) {
  install.packages("RODBC")
}

require (RODBC)

# Hent login fra personligt drev

if (file.exists("Y:/SQLkode.R")) {
  source("Y:/SQLkode.R")
} else {
  print("Der mangler log-in oplysninger!")}

# Log-in oplysninger ####
if (exists("Brugernavn") == FALSE) Brugernavn <- readline(prompt="Indtast Bru-
gernavn: ") #Tjekker om der tidligere er indtastet log-in oplysninger
if (exists("Kodeord") == FALSE) Kodeord   <-  readline(prompt="Indtast Kodeord:
") #Tjekker om der tidligere er indtastet log-in oplysninger

Server <- "devtest-oedb-sql16.vfltest.dk"
dbhandle <- odbcDriverConnect(connection=paste0("Driver={SQL Ser-
ver};server=",Server,";database=KundeAnalyseDB;trusted_connection=no;uid=",Brug
ernavn,";pwd=",Kodeord))

dfall <- sqlFetch(dbhandle, 'Region')
dfselect <- sqlQuery(dbhandle, 'select RegionName from Region where id > 1 ')

#http://www.statmethods.net/input/dbinterface.html

odbcClose(dbhandle)
```

Teknisk dokumentation

Eksempel på upload af MapInfo data til Sql-server (Powershell):

```
$user = "xxx"
$pass = "yyy"

$EasyLoader = "e:\FraRita\MapInfo\MapInfoEasyLoader\MapInfoEasyLoader.exe"

function MapInfoUpload($file, $table)
{
    $logfile = $PSScriptRoot + "\" + "$table" + (Get-Date).tostring("yyyyMMddhhmmss") +
".log"
    Write-Host $logfile
    & $EasyLoader /Q /R /T "$file;$table" /S "Driver={SQL Serv-
er};Server=$server;Database=KundeAnalyseDB;Uid=$user;Pwd=$pass;" /F $logfile | Out-Null
# Out-Null = Wait for window exit
}

MapInfoUpload "e:\FraRita\GHI2015\GHI2015.tab" "RH_GHI2015"
```

Download: http://www.pbinsight.com/support/product-downloads/for/easyloader
User Guide: http://reference1.mapinfo.com/software/easyloader/12.5/EasyLoaderUserGuide.pdf

perh@seges.dk

Brug af programmet CsvToSqlServer

CsvToSqlServer - CSV to Sql Table Utility.

This Program performs the following tasks, which can be switched on and off as required:

1) Analyze a CSV file and create a Sql table definition
   The CSV file can be optionally be fetched from a Url.
2) Create a database table
3) Upload a CSV file to a Sql table, replacing or adding data

Format: [-|--|/][name][=|:|][value]);

Parameters (text values):

F, filename	CSV filename (required)
W, url	Web url for CSV download
L, delimiter	Delimiter (default <tab>)
C, cultureinfo	CultureInfoName (default en-US)
E, encoding	Encoding of CSV file (default 1252)
T, table	Database Table Name
S, server	Sql Server Name
D, database	Database Name
U, user	Database User
P, password	Database Password
X, execute	Sql-file to execute after upload

Switches (turn on with ie -a or -a+, turn off with minus, ie -a-):

a, analyze	Analyze CSV (default true)
f, createsqlfile	Create Table Sql (default true)
c, createtable	Drop/Create Database Table (default false)
u, upload	Upload CSV To Database (default false)
t, truncate	Truncate Database Table before upload (default false)

Parameters and switches can be put in a text-file on the command line:

```
C:\CsvToSqlServer parameters.txt
```

Put each parameter on a separate line in the text file, ie

```
-F c:\mycsvfile.csv
–S mydatabaseserver
-D mydatabase
-T mytable
```

Note.

CultureInfoName styrer tal og dato format. Feks. er en-US tal i formatet 1,234.56 og da-DK er tal i formatet 1.234,56.

Encoding er den standard for talværdier der repræsenterer feks ÆØÅ som en tekstfil benytter. Encoding-værdier er typisk:

Nummer	Navn	Beskrivelse
850	ibm850	Western European(DOS) feks. Excel MS-DOS
1252	Windows – 1252	Western European(Windows) ANSI – normalt Windows format
28591	iso-8859-1	Western European (ISO) - latin1
65001	utf – 8	Unicode(UTF - 8)

Programmet detekterer normalt selv Unicode/UTF-8 format – I særlige tilfælde findes der dog UTF-8 filer uden type-identifikation (Byte Order Marks: BOM) i data og her må så anvendes 65001. Programmer som Notepad, Notepad2 og Notepad++ vil ofte kunne hjælpe med at finde det korrekte format.

Eksempel på udtræk fra Danmarks Statistik:

```
# http://api.statbank.dk/console
Write-Host ROOT "$($PSScriptRoot)\LoadPs1Config.ps1"

. "$($PSScriptRoot)\LoadPs1Config.ps1"

$dsid = "ANI51"
$csvfile = "$csvdir\$dsid.csv"

Write-Host Import $csvfile

$baseurl =
"http://api.statbank.dk/v1/data/$dsid/CSV?delimiter=Semicolon&DYRKAT=*&ENHED=*&Tid=*"
$response = Invoke-WebRequest -Uri $baseUrl
$table = $response.Content | ConvertFrom-Csv -Delimiter ";" # Convert to objects
$table1 = $table | Select-Object *,"MONTH","YEAR", "UpdateTime"  # Add extra columns

foreach($row in $table1) # Modify contents
{
    $row.YEAR = $row.TID.Substring(0,4)
    $row.MONTH = $row.TID.Substring(5,2)
    if ($row.INDHOLD -eq "..") { $row.INDHOLD = "" } # DS NULL indikator (!!)
            $row.UpdateTime = Get-Date -Format "yyyy-MM-dd hh:mm:ss"
}

$table1 | Export-Csv -Delimiter ';' -Path $csvfile -NoTypeInformation -Encoding UTF8

Write-Host Processor "$CsvToSqlServer"

& "$CsvToSqlServer" -F $csvfile -L ";" -C "da-DK" -S $server -D $database -T $dsid -U
$user -P $pw -cu

Exit $Error.Count #Signal errors
```

Ovenstående eksempel er et job der afvikles i et fast jobflow. Som det ses anvendes en konfigurationsfil der kan opdateres af Octopus i forbindelse med deployment.

perh@seges.dk

Ligeledes viser eksemplet hvordan de modtagne data kan modificeres i Powershell inden upload. Et alternativ ville være at definere dette som et ekstra Sql script der afvikles med –X parameteren efter upload.

Hvis data ikke skal regenereres Sql server, men tilføjes en bestående tabel anvendes følgende switch kombination: -afct-

Kildekoden findes i TFS-projektet OedbEksterneData.

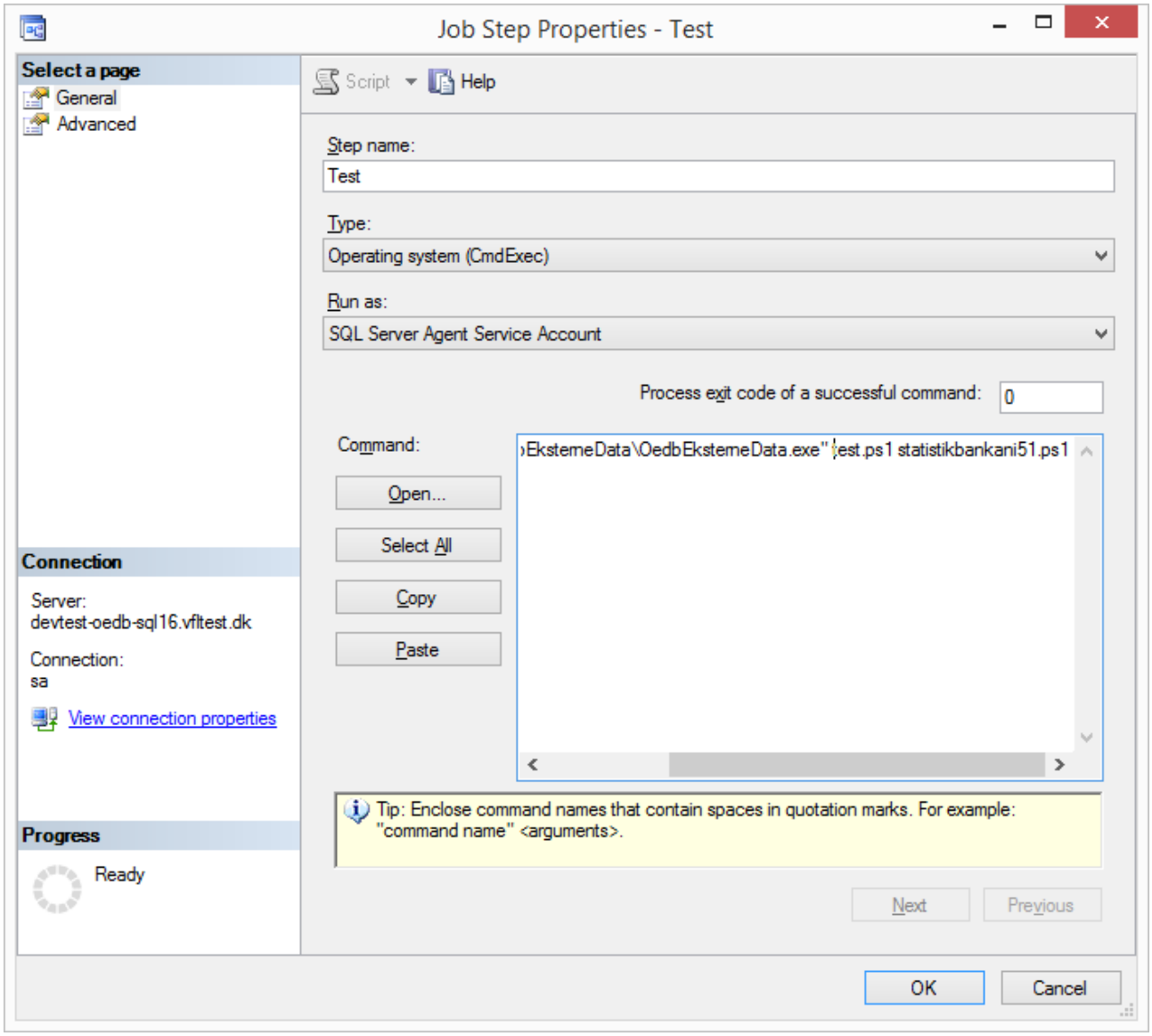
perh@seges.dk

Brug af programmet OedbEksterneData.

Dette program styrer afviklingen af udtræksjobs i Powershell, typisk ved at være scheduleret i Sql Servers job-agent (eller Windows Task Scheduler). Programmet registrerer udførelsen i OEDB loggen og danner samtidig en logfil som mail'es. Logning og mailing styres i programmets konfigurationsfil. Kildekoden findes i TFS-projektet OedbEksterneData.

Eksemplet her viser en afvikling hvor der køres 2 forskellige udtræk i et agent-job:

"c:\Program Files\OedbEksterneData\OedbEksterneData.exe" test.ps1 statistikbankani51.ps1



perh@seges.dk