Exreport

Inläsning av Primus-data till Exreport

Johan Ekman

Examensarbete för ingenjörs (YH)-examen
Utbildningsprogrammet för informationsteknik
Vasa 2018

EXAMENSARBETE

Författare:	Johan Ekman
Utbildningsprogram och ort:	Informationsteknik Vasa
Handledare:	Kaj Wikman

Titel: *Inläsning av Primus-data till Exreport*

Datum: 27.5.2018 Sidantal: 20

Abstrakt

Examensarbetet har utförts åt Vasa stad och uppdragsgivaren till examensarbetet är företaget Neotide Ab. Vasa stad använder Primus-systemet för att lagra information om bland annat elever, skolor, lärare och övrig personal. Tidigare har de manuellt skapat rapporter för exempelvis elevkostnader per skola. Dessa rapporter ska skapas på ett enklare sätt med hjälp av Neotides rapporteringssystem Exreport. Därmed är syftet med detta examensarbete att med hjälp av Exreport importera rådata från Primus-systemet till Exreport, för att kunden ska kunna använda Exreport för att skapa rapporter baserat på data från Primus-systemet.

Data från Primus exporteras till filer som automatiskt ska överföras en gång per dygn till Exreport-systemet. Data som behandlas är elevdata, lärardata, övrig personaldata samt skoldata. Felaktiga rader i inläsningsfilerna ska returneras via e-post till ansvarspersonerna på Vasa stad, så de själva kan gå igenom och förbättra sina rutiner. Resultatet blev att Exreport automatiskt läser in Primus-data till Exreport och man nu kan skapa rapporter baserat på dessa data i Exreport.

Språk: svenska Nyckelord: Primus, Python, Exreport

OPINNÄYTETYÖ

Tekijä: Johan Ekman
Koulutusohjelma ja paikkakunta: Tietotekniikka, Vaasa
Ohjaaja: Kaj Wikman

Nimike: Primus-tietojen importointi Exreportiin

Päivämäärä: 27.5.2018 Sivumäärä: 20

Tiivistelmä

Opinnäytetyö on tehty Neotide Oy:lle ja Vaasan kaupungille. Vaasan kaupunki käyttää Primus-järjestelmää tallentaakseen tietoa oppilaista, kouluista, opettajista ja muusta henkilökunnasta. Vaasan kaupunki on ennen tehnyt raportteja manuaalisesti Primuksen tiedoista, esim. koulukohtaisista oppilaskustannuksista. Asiakas haluaa luoda näitä raportteja Neotiden Exreport-raportointiohjelman avulla. Tämän opinnäytetyön tarkoitus oli tuoda Exreportin avulla tietoa Primus-järjestelmästä Exreportiin.

Tieto Primuksesta tuodaan Exreportiin kerran päivässä. Tietoa, jota tullaan käsittelemään, on oppilas-, opettaja-, koulu- sekä muuta henkilökuntatietoa. Virheellisiä rivejä tiedostoissa palautetaan Vaasan kaupungin tämän opinnäytetyön yhteyshenkilöille , jotta he voivat käydä ne läpi. Tulos oli, että Exreport automaattisesti tuo Primus-tietoa Exreportiin ja asiakas voi nyt Exreportin avulla luoda raportteja näistä tiedosta.

Kieli: ruotsi Avainsanat: Primus, Python, Exreport

BACHELOR'S THESIS

Author:	Johan Ekman
Degree Programme:	Information Technology, Vasa
Supervisor:	Kaj Wikman

Title: Importing Primus Data to Exreport

Date: May 27, 2018 Number of pages: 20

Abstract

This Bachelors Thesis is conducted for the city of Vaasa, which is a customer of the company Neotide Oy. The customer uses the Primus system to save information about students, schools, teachers and other staff. Prior to this thesis, the city of Vaasa has created such reports manually, when they, for instance, want to calculate student costs for schools. Consequently, they wish to use Neotide's reporting system Exreport to create these reports in an easier way. The purpose of this thesis was to import raw data from the Primus system to Exreport. Accordingly, the customer will be able to use Exreport to create reports based on the data from Primus.

The Primus data will be imported once a day to the Exreport system. Data consist of information about students, schools, teachers and other staff. Faulty data are returned to the customer for examination. The result is that Exreport automatically imports Primus data to Exreport and that the customer can create reports in Exreport based on these data.

Language: Swedish Key words: Primus, Python, Exreport

Innehållsförteckning

1 Inledning

I detta kapitel presenteras examensarbetet bakgrund och syfte samt företaget Neotide Ab.

1.1 Bakgrund

Detta examensarbete genomförs åt företaget Neotide och uppdragsgivaren är Vasa stad. Vasa stad har länge varit i behov av att utveckla och förbättra sina rutiner när det kommer till att skapa rapporter på data de har om bland annat elever och lärare i sina grundskolor. Rapporter som de brukar skapa redovisar till exempel kostnader per elev och elevantal i skolorna.

Vasa stads skolor använder Primus-systemet för att spara information om skolor, elever, lärare och övrig personal. Hittills har Vasa stad manuellt samlat data från Primus-systemet som lagrats i olika Microsoft Excel-filer. Med hjälp av verktygen i Excel har de skapat olika rapporter och grafer som de behöver för att planera kommande läsår och budget. Samtidigt kan de med hjälp av dessa rapporter göra en bedömning av sin verksamhet och se vad de kan förbättra.

Denna rapporteringsprocess har Vasa stad bett Neotide förbättra och automatisera. Eftersom Vasa stad redan använder Neotides rapporteringssystem Exreport vill de använda samma system för de nya rapporterna. Tanken är att en gång per dygn ska filer skickas med data från Primus-systemet, som automatiskt läses in till Exreport för rapportering.

1.2 Neotide

Neotide Ab är ett företag som grundades 1999 i Helsingfors av Johan Kullas och Patrik Simons. År 2001 flyttade företaget till Vasa och har varit beläget i Vasa sedan dess. Företaget har i dagsläget 15 anställda och företagets huvudsakliga uppgift är att utveckla mjukvaror åt städer, kommuner och företag i hela Finland. De huvudsakliga kunderna är sjukhus och kommuner. (Neotide Ab, 2018.)

Neotide har skapat flera olika program, men det viktigaste programmet som kommer att användas i detta examensarbete är Exreport. Exreport är ett rapporteringsprogram som kombinerar data från olika databaser och kopplar ihop data med nycklar för att man på ett enkelt sätt ska kunna skapa rapporter enligt eget tycke.

1.3 Syfte

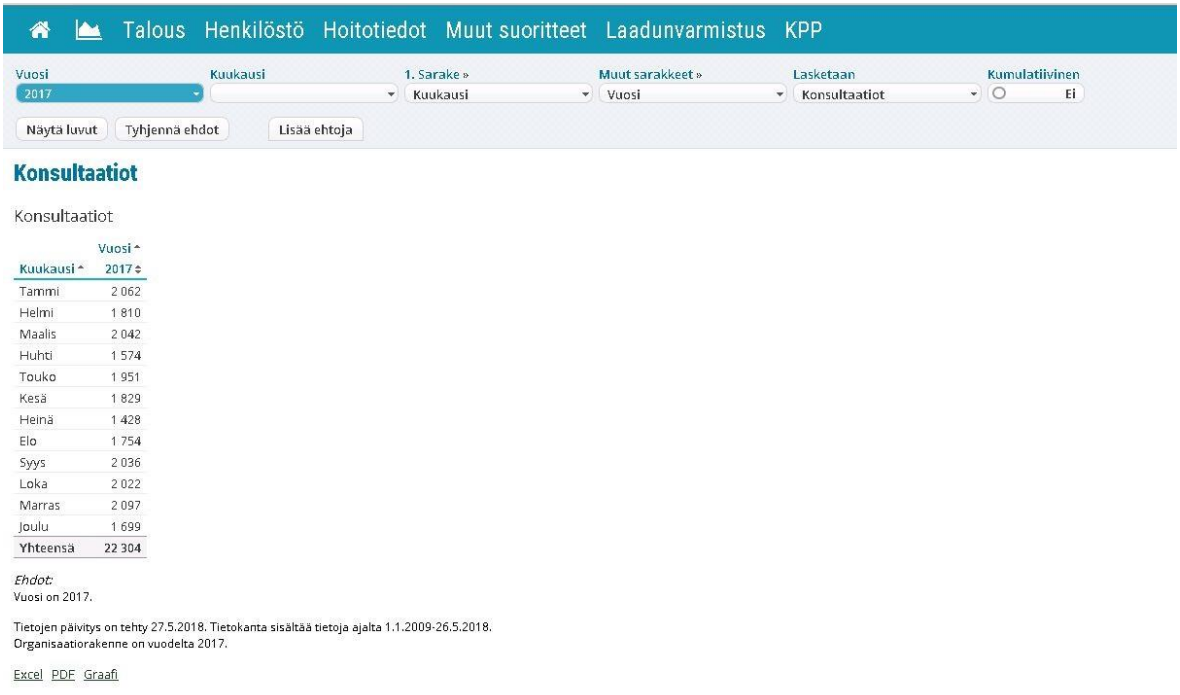
Syftet med detta examensarbete är att importera rådata från Primus-systemet till Exreport. Mer specifikt innebär detta att kunden ska kunna skapa rapporter med Exreport baserat på data som blir inläst från Primus-exportfilerna. Examensarbetet avgränsas till att redogöra för inläsningen av Primus-filer i praktiken och samtidigt visa exempel på rapporter gjorda i Exreport på de inlästa filerna.

2 Teori och teknik

I det följande presenteras rapporteringsprogrammet Exreport, företaget Visma, utvecklingsmiljön Jetbrains Pycharm Community Edition, programmeringsspråket Python, filformatet ZIP, databashanteringsprogrammet Microsoft Sql Server Management Studio som används i detta examensarbete. Annat som tas upp är databaser, normalisering av data samt importering av filer.

2.1 Exreport

Exreport är ett rapporteringsprogram som företaget Neotide Ab säljer. Med hjälp av Exreport kan man till exempel skapa rapporter (se figur 1) baserat på ekonomi- och patientuppgifter. Programmet nås via en webbläsare och är speciellt inriktat informationsbehovet inom hälsovården. Informationen i rapporterna är baserad på data lagrat i Exreports egna databas. Exreport hämtar data från de ursprungliga källorna vid tidsbestämda hämtningar. (Neotide Ab, 2018.)



Figur 1. Exempel på en rapport skapad i Exreport.

2.2 Visma

Visma är ett företag som säljer programpaketet *Visma InSchool*, vilket innehåller programmen Primus, Wilma och Kurre. Detta programpaket är speciellt inriktat på skolundervisning och -administration. (Visma, 2018.) Antalet skolor som använder Visma i Finland lär vara stort, men dessvärre finns ingen exakt statistik att tillgå.

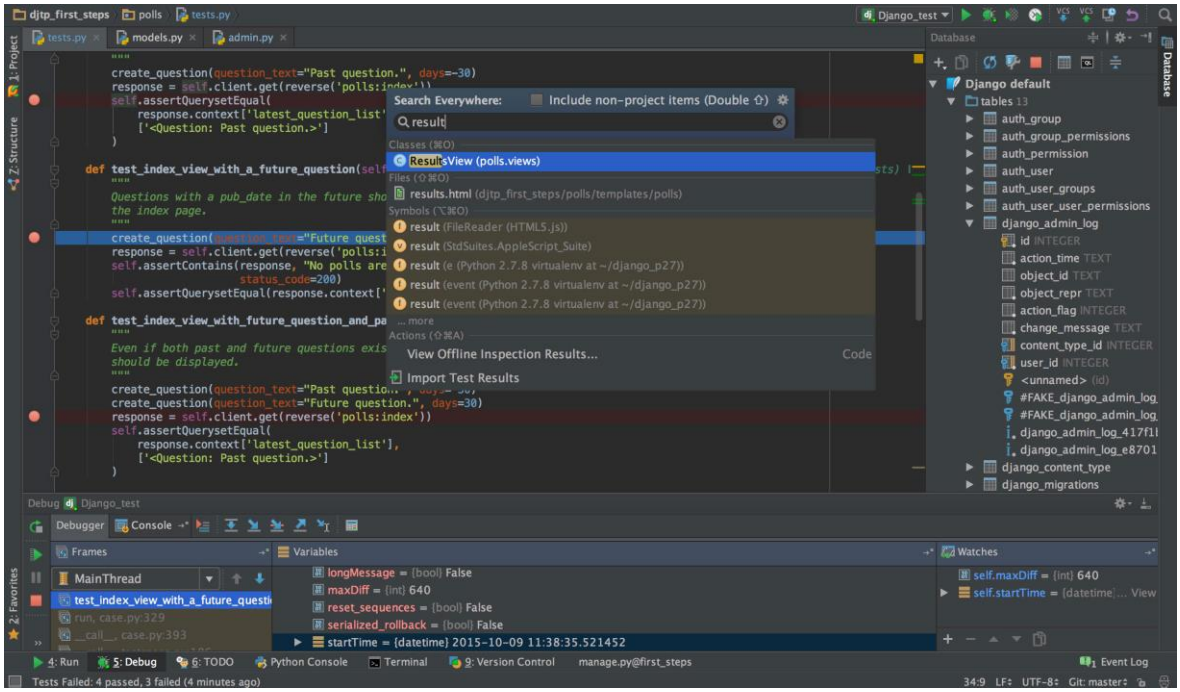
Primus är ett register där administration och lärare sparar dagligt arbete och planering. Programmet kan skräddarsys enligt kundens behov och innehåller information om elever, lärare och personal, samt kurser, vitsord, certifikat och skolskjutsar. (Visma, 2018.)

I *Wilma* kan elever och vårdnadshavare kommunicera med lärare och övrig personal, kontrollera sin egen frånvaro och skriva ut scheman samt kursvitsord. Vårdnadshavare har utöver det möjlighet att motivera och styrka elevers frånvaro samt meddela om kommande frånvarotillfällen. Systemet har olika meddelandefunktioner, såsom möjligheten att skicka meddelanden direkt till mobila apparater. Med hjälp av detta strävar man efter att underlätta kommunikationen mellan hemmen och skolan. (Visma, 2018.)

I *Kurre* skapas arbetsscheman, kommande läsår planeras och räknandet av lärarnas och elevernas arbetstimmar utförs. Programmet är helt och hållet integrerat i Primus, men finns också tillgängligt separat. Kurre är lämpligt för skolor med flera undervisningsenheter och samarbete mellan andra stadiets skolor. Gemensamma utrymmen som samarbetsskolorna har till förfogande samt scheman för personal som jobbar åt alla samarbetande skolor syns för alla samarbetsskolor och tack vare det kan krockar i scheman förhindras. Samma gäller lån av maskiner och arbetsredskap. (Visma, 2018.)

2.3 Jetbrains Pycharm

Jetbrains Pycharm (se figur 1) är designat av programmerare för programmerare. Pycharm erbjuder de verktyg som behövs för att på ett så effektivt sätt som möjligt utveckla system med programmeringsspråket Python. Med Pycharm är det lätt att skriva lättläst kod samtidigt som IDE:n, vilket står för Integrated Development Evironment, hjälper att kontrollera koden så den uppfyller alla stilkrav som finns i Python. (JetBrains, 2018.)



Figur 2. Användargränssnittet i Pycharm (JetBrains, 2018).

2.4 Python

Python är ett interaktivt, objektorienterat programmeringsspråk jämförbart med Perl, Ruby, Scheme och Java. Python är ursprungligen skapat av Guido van Rossum. Programmeringsspråket har fått sitt namn från den kända serien ”Monty Python”. Van Rossum ville ha ett kort, unikt och mystiskt namn på sitt programmeringsspråk, därav ansåg han benämningen Python vara utmärkt. Det kostar ingenting att ladda ner eller använda Python, inte heller att använda det i egna applikationer. Python är portabelt och kan köras på många unix-varianter, på Mac samt på Windows 2000 eller senare operationssystem. Python är också fritt att modifiera. (Python Software Foundation, 2018.) Exempel på hur en klass skriven i Python kan se ut fås i figuren nedan (figur 3).

```
class MyClass:
    """A simple example class"""
    i = 12345

    def f(self):
        return 'hello world'
```

Figur 3. Exempel på klass i Python (Python Software Foundation, 2018).

Om det skrivna programmet blir längre finns det eventuellt behov för att dela upp det i flera olika filer för att underlätta underhållningen av programmet. Det kan även finnas behov av att använda en funktion i flera olika program utan att behöva kopiera in funktionens definition i varje program. För att stöda detta kan funktionerna placeras i en fil, en så kallad modul (se figur 4). Funktioner från denna modul kan sedan användas i andra filer och program genom att importera modulen (se figur 5).

```
# Fibonacci numbers module

def fib(n):    # write Fibonacci series up to n
    a, b = 0, 1
    while b < n:
        print(b, end=' ')
        a, b = b, a+b
    print()

def fib2(n):   # return Fibonacci series up to n
    result = []
    a, b = 0, 1
    while b < n:
        result.append(b)
        a, b = b, a+b
    return result
```

Figur 4. Exempel på modul i Python (Python Software Foundation, 2018).

```
import fibo
```

Figur 5. Exempel på hur man importerar modulen "fibo" för att kunna använda dess funktioner (Python Software Foundation, 2018).

2.5 ZIP

En fil med ändelsen .zip är en ZIP-komprimerad fil och är det mest vanliga filformatet för arkivering. ZIP-formatet, likt andra filformat för arkivering, är enkelt förklarat en samling av en eller flera filer som är komprimerade in i en enda fil för att enkelt kunna flytta filerna och för att filstorleken ska vara mindre. ZIP-filer används oftast till för nedladdning av mjukvara. Genom att "zippa" mjukvaroprogrammet sparas utrymme på servern, nedladdningstiden blir kortare och filerna hålls snyggt och organiserade i en enda ZIP-fil.

Ett annat exempel är vid nedladdning eller delning av en stor mängd bilder. I stället för att skicka alla bilder individuellt med e-post eller spara alla bilder individuellt från en

hemsida, kan avsändaren placera alla bilder i en ZIP-fil för att endast en fil ska behöva flyttas. De flesta operativsystem, både Microsofts och Apples operativsystem, har inbyggd programvara för att kunna hantera ZIP-filer. (Tim Fisher, 2018.)

2.6 Microsoft Sql Server Management Studio

Microsoft Sql Server Management Studio är en integrerad databasmiljö som hanterar olika infrastrukturer, från SQL Server till SQL databaser. Programmet erbjuder olika verktyg för att konfigurera, övervaka och administrera olika instanser av SQL. (Microsoft, 2018.) SQL står för Structured Query Language och är ett programmeringsspråk designat för att hantera data som har sparats i relationsdatabaser. SQL körs genom enkla kommandon, så kallade frågor. Tack vare simpliciteten i frågorna hålls data säkert och integriteten hålls hos databasen, oavsett dess storlek. Med Sql Server är det enkelt att köra Sql-frågor (Codeacademy, 2018.) För att hämta data från en tabell används en select-sats (se figur 6) och för att uppdatera värdet i en eller flera kolumner används en update-sats (se figur 7). För att lägga till en rad till en tabell används insert (se figur 8) och för att ta bort en eller flera rader från en tabell används delete (se figur 9).

```
SELECT column_name FROM table_name;
```

Figur 6. Exempel på en select-sats (Codeacademy, 2018).

```
UPDATE table_name
SET some_column = some_value
WHERE some_column = some_value;
```

Figur 7. Exempel på en update-sats (Codeacademy, 2018).

```
INSERT INTO table_name (column_1, column_2, column_3) VALUES (value_1,
'value_2', value_3);
```

Figur 8. Exempel på en insert-sats (Codeacademy, 2018).

```
DELETE FROM table_name WHERE some_column = some_value;
```

Figur 9. Exempel på en delete-sats (Codeacademy, 2018).

2.7 Databaser

Exreport använder sig av data sparat i sin egna databas för att skapa rapporter. Eftersom Vasa stad redan använder Exreport behövdes ingen ny databas för projektet. För att alla filer skulle kunna läsas in till Exreports databas behövdes tabeller för Primus-data. För att kunna lägga till tabeller måste normalisering av data utföras. Utan det skulle det vara onödigt arbete att först skapa några tabeller och läsa in data och sedan upptäcka att tabellerna innehåller redundant data och är i behov av normalisering.

2.8 Normalisering av data

Normalisering av data innebär att en databas organiseras i tabeller och kolumner. Meningen är att en tabell ska handla om ett visst ämne och att endast kolumner med data som hör till samma ämne finns i tabellen. Genom att begränsa innehållet i en tabell till att endast innehålla relevant information om ämnet förhindrar man samma värde från att förekomma flera gånger i tabellen. I och med normaliseringen sparas med andra ord utrymme i databasen. Det finns tre huvudorsaker varför man bör normalisera en databas. För det första minimerar man förekomsten av duplicerade data, för det andra undviker man problem med att modifiera data och för det tredje blir Sql-frågor enklare och tydligare. (Easy Computer Academy, LLC, 2018.)

2.9 Importering av filer

Alla filer som ska importeras dagligen till Exreports databas står i filformatet Comma Separated Values (CSV). CSV är ett vanligt filformat som används för importering och exportering av data mellan kalkylark och databaser. (Python Software Foundation, 2018.) För att importera filerna användes pythons csv-modul. Csv-modulen innehåller funktioner för att exempelvis läsa från eller skriva till csv-fil. Eftersom filer ska läsas *från* i det här examensarbetet används csv-modulens reader-funktion (se kodexempel 1).

Kodexempel 1. Exempel på användningen av reader-funktionen i pythons csv-modul.

```
import csv
with open('eggs.csv', newline='') as csvfile:
    spamreader = csv.reader(csvfile, delimiter=' ', quotechar='|')
    for row in spamreader:
        print(', '.join(row))
```

3 Genomförande

I detta kapitel presenteras arbetets planering samt hur arbetet utfördes.

3.1 Planering och specifikationer

Planeringen inleddes med ett möte med representanter för Vasa stad i maj 2017. Under detta möte gick vi igenom vad som ska göras och hur det ska göras. I detta examensarbete kommer jag fokusera på de viktigaste kraven som ställdes. Jag kommer även att visa ett exempel på en rapport skapad med hjälp av Exreport, baserat på data som blivit inläst från Primus-filerna.

Kraven och specifikationerna jag fokuserar på i examensarbetet är följande:

- Varje dag klockan 16:00 exporteras filer från Primus till en mapp på Vasa stads server. Dessa filer ska läsas in i Exreports databas före de blir överskrivna nästa dag, samma tid.

- Utöver att vi ska läsa in filerna till Exreport ska vi även arkivera dessa filer för att kunna gå bakåt i tiden och kontrollera en viss tidpunkt. För att inte fylla lagringsutrymmet mer än nödvändigt bör dessa filer komprimeras. I framtiden är det meningen att man ska kunna gå bakåt i tiden i Exreport och kontrollera exempelvis elevantal vid ett visst datum för skolorna, men det behövs inte i dagsläget.

3.2 Kopiering av filer

Alla filer som exporteras från Primus-systemet till en mapp på Vasa stads server har samma namn varje dag, detta betyder i praktiken att filnamnen inte innehåller någon information om när filerna är skapade. För att hålla filerna organiserade valde jag att i kopieringsskedet lägga till kopieringens tidsstämpel i slutet av filnamnet. Exempelvis filen "PrimusToAdOpettajatiedot.csv" får istället namnet "PrimusToAd-Opettajatiedot201805170301.csv" (se figur 8). Den nya filen kopieras till en annan mapp på serverns hårddisk. I teorin kan man ha kvar filerna i samma mapp, men för säkerhets skull kopieras de till en helt annan mapp (se kodexempel 2), vilket underlättar arbetet med att separera de nyaste filerna från de äldre. För att säkerhetsställa att rätt filer kopieras kontrolleras filnamnet mot förutbestämda filnamn i en lista.

PrimusToAdArviointikirjat.csv --> PrimusToAdArviointikirjat201805170301.csv
PrimusToAdHenkilokuntatiedot.csv --> PrimusToAdHenkilokuntatiedot201805170301.csv
PrimusToAdKouluntiedot.csv --> PrimusToAdKouluntiedot201805170301.csv
PrimusToAdOppilastiedot.csv --> PrimusToAdOppilastiedot201805170301.csv
PrimusToAdOpettajatiedot.csv --> PrimusToAdOpettajatiedot201805170301.csv

Figur 8. Figur över filernas tidigare namn utan och med tidsstämpel.

Kodexempel 2. Kodexempel på funktion för att kopiera filerna.

```
def copy_files(self):
    dest_path = os.path.join(self.source_path, 'files', 'primus')
    os.makedirs(dest_path, exist_ok=True)
    files = []
    now = datetime.datetime.now()
    for filename in os.listdir(self.file_directory):
        name, ext = os.path.splitext(filename)
        new_filename = os.path.join(dest_path, name + now.strftime("%Y%m%d%H%M") + ext)
        files.append((filename, new_filename))
        shutil.copy(os.path.join(self.file_directory, filename), new_filename)
    return files
```

3.3 Arkivering av filer

Filerna kan inte sparas i ursprungsformatet på servern för då används onödigt lagringsutrymme på servern. Python har en färdig modul för att skapa, läsa och skriva ZIP-filer. I och med skapandet av ZIP-filen komprimeras filerna och slutliga filstorleken är en

bråkdel jämfört med okomprimerat format. Precis som med de kopierade filerna läggs en tidsstämpel till i namnet på ZIP-filen för att hålla allting organiserat (se kodexempel 3).

Kodexempel 3. Kodexempel på funktionen för arkivering av filerna.

```
def archive(self, files):
    now = datetime.datetime.now()
    zip_path = os.path.join(self.source_path, 'files', 'primus', 'primus%s.zip' % now.strftime("%Y%m%d%H%M"))
    with zipfile.ZipFile(zip_path, 'w', compression=zipfile.ZIP_DEFLATED) as z:
        for filename, path in files:
            z.write(path, os.path.split(path)[-1])
    for filename, path in files:
        os.remove(path)
```

3.4 Importerade filer

I detta kapitel skrivs det om hur de importerade filernas utseende samt exempel på kod som användes vid importeringen.

3.4.1 Lärare och övrig personal

En fil som importerades innehåller data om lärarna. Redan vid en första anblick på filen (se figur 10) upptäcktes ett problem: Varje lärare i filen har endast en rad i filen fastän en lärare ofta undervisar i flera olika ämnen. Fältet som beskriver ämnet innehåller alla ämnen som läraren har, separerat med '#'-tecken.

```
2017-2018;3317;MMT#S2;6#6;;;;;4038#4038#4038;3317#3317#3317;345#345#345;31051#31052#31052;6#2#4;12;Ei;
2017-2018;3340;OPO;;8,5;;#TVA KE;21,5;4000#4000;3340#3340;342#342;31032#31037;8,5#21,5;30;Ei;
```

Figur 10. Exempelrader ur filen med lärardata.

Kolumnerna som efterföljer ämneskolumnen definierar hur många timmar läraren har för detta ämne i olika kategorier: timmar för grundskolans lägre och högre årskurser eller specialundervisning. För att kunna importera raderna från lärarfilen behövdes en programmeringskod för att separera värdena med ”#”-tecknet (se kodexempel 5). För att underlätta detta skapades en funktion som summerar ihop timmarna per ämne och timtyp (se kodexempel 4). Med hjälp av denna programmeringskod skapades flera rader i tabellen för lärarnas ämnen respektive timantal.

Kodexempel 4. Kodexempel på funktion som används för att summera timmar per ämne.

```
def get_tunnit(ala, yla, eri, aine):
    tunti_dict = collections.defaultdict(float)
    for index, aine_values in enumerate(aine):
        for tyyppi, tt in (('ala', ala), ('yla', yla), ('eri', eri)):
            if tt[index]:
                tunti_dict[(aine_values, tyyppi)] += float(tt[index].replace(',', '.'))
    return tunti_dict
```

Kodexempel 5. Kodexempel på en del av koden som använder get_tunnit()-funktionen.

```
aine_list = line[7].split('#')
aine_list_length = len(aine_list)
ala_list = ((line[8] + '#' * aine_list_length).split('#'))
yla_list = ((line[9] + '#' * aine_list_length).split('#'))
eri_list = ((line[10] + '#' * aine_list_length).split('#'))
tuntilist = get_tunnit(ala_list, yla_list, eri_list, aine_list)
for (aine_value, tuntityyppi), tunnit in tuntilist.items():
    obj_aine.values['tuntityyppi'] = tuntityyppi
    obj_aine.values['aine'] = aine_value
    obj_aine.values['tunnit'] = tunnit
    obj_aine.save(cursor)
```

Lärardata finns i en fil, men eftersom normalisering inte är gjord görs normaliseringen allra först. Efter noggrann undersökning av filen kom jag fram till att data består egentligen av tre olika delar: En del är information om läraren, en om lärarens ämnen och en del hör till bokföringen av lärarens timmar. Således blir en rad i lärarfilen importerad till tre olika tabeller som skapades i Exreports databas.

Detta resulterade i en huvudtabell som döptes till ”PrimusOpettaja” där det finns information som används för att beskriva en lärare. Informationen innehåller personbeteckning, läsår, skola, och det totala timantalet. Med timantal avses lärarens totala antal timmar i alla ämnen. Andra tabeller som lärardata blir uppdelat i döps till ”PrimusOpettajaAine” och ”PrimusTiliointi”. PrimusOpettajaAine innehåller en rad för varje ämne som läraren undervisar i. För varje ämne fylls det totala timantalet i olika kolumner för lågstadie-, högstadie- och specialundervisningstimmar. PrimusTiliointi innehåller en rad med totala timmar per ”tili”, ”tulosyksikkö”, ”tehtävä” och ”kohde”.

Anledningen till denna indelning är att timantalet per ämne inte anger specifikt vilken skola undervisningen sker på. I praktiken kan en lärare undervisa i flera olika skolor, och då är det viktigt att i bokföringen hålla koll på hur mycket kostnader olika skolor står för.

Anledningen till att ”PrimusTiliointi” inte innehåller ordet ”Opettaja” är att i den tabellen kommer även data från filen med data om övrig personal. Övrig personals timmar är inte uppdelade per ämne i exportfilen, utan den innehåller mest data om hur timmarna bokförs. För att kunna ha två olika typer av personer lades en tyyppi-kolumn till som definierar hurudan person det är fråga om. Orsaken varför det görs så är för att enkelt kunna separera lärares och övrig personals timmar.

I denna tabell läggs också kolumnen ”tyyppi” till. Eftersom det ska vara möjligt att separera lärartimmar från övrig personals timmar i Exreport behövs en kolumn som beskriver vad det är för typ av person. En annan orsak till att det behövs en kolumn som beskriver personens typ är att en lärare inte nödvändigtvis bara är en lärare; hen kan också höra till övrig personal. För att inte ta upp mycket utrymme i tabellen kan denna kolumn vara av datatypen ”bit”. ”Bit” är en datatyp som endast innehåller antingen värdet 0 eller 1. Det enda som bör beaktas i rapporteringsskede och i importeringsskede är vilken som är vilken.

3.4.2 Elevdata

Elevdata är en fil med information om eleven. Information som finns om eleven är exempelvis personbeteckning, läsår, organisationsenhet, årskurs, klass och övriga kolumner som informerar om eleven till exempel är i behov av specialundervisning. Varje elev har ett eget unikt ID, som kommer från ett landsomfattande system. Varje elev i hela landet har ett ID som består av sifferkombinationer. Ibland kan elever utan ID dyka upp i elevfilen som kommer från Primus. Detta beror på att Vasa stad har elever i sitt system som har Vasa som hemkommun, men går i skolan i någon annan kommun. Då syns inte detta ID. Sådana elever ska inte heller Vasa stad kunna rapportera i detta skede, och därför ska de inte läsas in i databasen.

3.4.3 Skoldata

Skoldata är en fil med information om skolorna. Kolumner som finns med är exempelvis resultatenhet, läsår, hyra, grundundervisningsklassernas antal, elevantal. Elevantalet är ett antal som bestäms 20 september varje läsår. Detta antal är ett medeltal av de två föregående läsåren.

3.4.4 Personaldata

Liksom lärardata innehåller personaldata personbeteckning, läsår, resultatenhet, timantal. Skillnaden på lärardata och personaldata är att personaldata innehåller information om kuratorer, hjälplärare osv. Personalens timmar är uppdelade enligt samma modell som lärarnas tiliointi-del, och timmarna räknas som årsveckotimmar. Därför väljer jag att läsa in de timmarna i samma tiliointi-tabell för att på ett enkelt sätt kunna välja i rapporten om man vill rapportera lärare, personal eller båda två på samma gång.

3.5 Hjälptabell

Alla filer som innehåller information om personer har personbeteckning samt namn. I början av projektet var både personbeteckning och namn en del av persontabellernas nycklar, men i och med att rader med felaktiga personbeteckningar inte importeras behövs inte personernas namn i flera tabeller utan namnen kunde flyttas till en särskild hjälptabell för detta. Elevdata, lärardata och personaldata innehåller utöver personbeteckning även personernas fullständiga namn. För att smidigt kunna begränsa vem som har tillgång till namn och personbeteckning för olika resultatenheter skapas en separat tabell där information om personbeteckning och namn samt vad det är för typ av person lagras. Nyckel i detta fallet är personbeteckning som kopplas ihop med andra tabeller.

3.6 Problem vid inläsning

När filerna undersöktes i projektets början upptäcktes det att exempelvis lärarnas timmar inte alltid är heltal utan de kan även vara decimaltal med kommatecken i. Eftersom CSV-filer som standard separerar värden med kommatecken innebar detta att utan att ändra på tecknet som separerar fälten i filen skulle det vara omöjligt att avgöra ifall det rör sig om ett decimaltal eller ifall det handlar om ett nytt fält. Därför ombeddes kunden ändra på tecknet som separerar fälten till ett semikolon.

Filerna som importeras till Exreports databas kan innehålla en del felaktiga rader. Exempel på detta är elever som i importfilen saknar en giltig personbeteckning eller det nationella unika ID som varje elev ska ha. Alla sådana fall sparas i en lista som innehåller födelsetid, efternamn samt skola. Anledning till att endast födelsetid används är för att personens personbeteckning är känsliga data som absolut inte får läcka ut. Det samma gäller för alla motsvarande fall i alla andra filer som importeras. Listan läggs sedan in i en excel-fil (se kodexempel 6) som skickas via e-post till två personer på Vasa stad samt till min egna e-

post. Detta görs för att på ett enkelt sätt få en överblick över vilka rutiner som ännu måste förbättras från Vasa stads sida samt för att inte felaktigt data ska komma in i tabellen.

Kodexempel 6. Kodexempel på funktion för att skapa excel-fil

```
def make_excel(self, missing_id_list):
    book = openpyxl.Workbook()
    sheet = book.active
    row = 1
    for item in missing_id_list:
        if isinstance(item, list):
            for i, value in enumerate(item):
                sheet.cell(column=i + 1, row=row, value=value)
            row += 1
            continue
        sheet.cell(column=1, row=row, value=item)
        row += 1
    string = BytesIO()
    book.save(string)
    string.seek(0)
    return string.read()
```

4 Resultat

Resultatet av detta examensarbete är att Exreport automatiskt kopierar de exporterade filerna från Primus, arkiverar filerna för att enkelt i framtiden kunna kontrollera hur exempelvis elevsituationen såg ut vid ett visst tillfälle, samt importerar data från filerna till Exreport. Ifall filerna innehåller rader som exempelvis har felaktiga personbeteckningar så importeras dessa rader inte till Exreport databas. De raderna returneras istället som en lista till ansvarspersoner på Vasa stad som då har möjlighet att gå igenom listan och korrigera raderna i källsystemet så att de raderna kan importeras till Exreport. Med hjälp av Exreport kan kunden själv skapa en del av de rapporter som de tidigare skapat manuellt.

Designen behövdes inte skapas, eftersom Exreport i sig är ett färdigt rapporteringsprogram med egen design. En design som kunden är van med att använda när de redan har Exreport i bruk för andra delar än att rapportera från primus. Ett exempel på en rapport (se figur 11) som nu kan skapas med hjälp av Exreport är kostnader per elev.En rapport över kostnader per elev kan se ut på många olika sätt och i detta exempel så är kostnaderna uppdelade i olika uppgifter. En uppgift är något som används i bokföringen, och är ungefär som en kategori som informerar om till vilken kategori kostnaderna hör.

Figur 11. Exempel på hur kostnaderna per elev kan se ut för en skola.

5 Diskussion

Sett till helheten är jag nöjd med resultatet av detta examensarbete. Tanken var först att arbetet inte skulle bli något jag skriver examensarbete om men ju mer projektet växte och när jag märkte hur mycket som behövdes tas i beaktan för att alls kunna läsa in data från Primus så ansåg jag det vara ett ytterst lämpligt examensarbete.

En del av arbetet skulle jag ha kunnat göra annorlunda. Till exempel skulle jag kunnat planera lite mera förrän jag började på med det praktiska arbetet. Eftersom att det fanns många faktorer som påverkade bland annat hur tabellstrukturen kunde se ut och att många av de faktorerna upptäcktes först då man jobbat en längre tid med projektet så skulle extra planering inte hjälpt nämnvärt.

Vidareutveckling av projektet är på gång, bland annat så finns det en del kolumner och fält som kommit till i efterhand. I filen med elevdata finns nu också en kolumn med datum och det datumet är när eleven börjat på den skolan eleven är på nu. Denna kolumn finns i Primus men uppdateringen av den har ibland inte skett och därför kan det i dagsläget finnas dubbletter i databasen. Skulle det ha varit möjligt att få direkt tillgång till källan varifrån data kommer så skulle man kanske ha kunnat se vilka olika kolumner som faktiskt är nycklar i tabellerna.

Eftersom Primus används i skolor över hela Finland är mina förhoppningar att fler städer vill köpa resultatet av detta examensarbete. Ifall någon annan köpare hittas återstår det att se hur mycket som skiljer sig från stad till stad. Primus kan användas på lite olika sätt men i grund och botten borde det inte finnas stora skillnader mellan olika skolor.

I och med att detta var mitt första stora projekt på Neotide är det just detta examensarbete som har lärt mig det mesta jag kan om Python. I princip hade jag inte alls programmerat i detta programmeringsspråk förrän jag började med detta projekt.

En annan sak jag fått lära mig mycket mer om som jag inte haft så mycket erfarenhet av i praktiken är normalisering av data. Tidigare har jag nog läst in data i tabeller men inte behövt beakta formatet på data som jag behövt i detta examensarbete. Jag har stor nytta i mitt dagliga arbete då jag fått göra detta projekt.

Källförteckning

Codeacademy, 2018. List of SQL commands. [Online]
https://www.codecademy.com/articles/sql-commands [hämtat: 15.05.2018]

Easy Computer Academy, LLC, 2018. Database Normalization Explained in Simple English. [Online]
https://www.essentialsql.com/get-ready-to-learn-sql-database-normalization-explained-in-simple-english/ [hämtat: 14.05.2018]

JetBrains, 2018. Python IDE for Professional Developers. [Online]
https://www.jetbrains.com/pycharm/ [hämtat: 21.05.2018]

Microsoft, 2017. Download SQL Server Management Studio (SSMS). [Online]
https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms
[hämtat: 2.10.2017]

Neotide Ab, 2018. Exreport - Utökad Rapportering. [Online]
http://neotide.fi/exreport_sv.html [hämtat: 27.05.2018]

Neotide Ab, 2018. Företaget. [Online]
http://neotide.fi/foretaget.html [hämtat: 30.01.2018]

Python Software Foundation, 2018. Classes. [Online]
https://docs.python.org/2/tutorial/classes.html [hämtat: 21.05.2018]

Python Software Foundation, 2018. CSV File Reading and Writing. [Online]
https://docs.python.org/3/library/csv.html [hämtat: 16.05.2018]

Python Software Foundation, 2018. General Python FAQ. [Online]
https://docs.python.org/2/faq/general.html [hämtat: 21.05.2018]

Python Software Foundation, 2018. Modules. [Online]
https://docs.python.org/3/tutorial/modules.html [hämtat: 21.05.2018]

Tim Fisher, 2018. What is a ZIP file? [Online]
https://www.lifewire.com/zip-file-2622675 [Hämtat: 26.05.2018]

Visma, 2018. Kurre. [Online]
https://www.visma.fi/inschool/kurre/ [hämtat: 30.01.2018]

Visma, 2018. Primus. [Online]
https://www.visma.fi/inschool/primus/ [hämtat: 30.01.2018]

Visma, 2018. Wilma. [Online]
https://www.visma.fi/inschool/wilma/ [hämtat: 21.05.2018]

Figurförteckning

Förteckning över kodexempel