Forsyningsdigitaliseringsprogrammet

Tværgående Arkitektur og Udvikling (TAU) Leverance - Fælles retningslinjer for dataformater for deling af forbrugsdata

Udarbejdet: TAU arbejdsspor 2. vedr. interoperabilitet

Medlemmer at arbejdsgruppe:
Niels Ejnar Helstrup Jensen, Senioringeniør, Energinet

Bruno Alexander Højrizi, Leder af GIS center, Aalborg Forsyning

Steen Grube, Tidl. Head of Department, Tidl. HOFOR

Steen Kramer, Senior specialist, Dansk Fjernvarme

Andreas Okkels Overgaard Thomsen, Senior Business Developer, Center Denmark

Maya Borges, Dataarkitekt, Digitaliseringsstyrelsen

Helle Pernille Hansen, Fagleder, Aarhus Vand

Peter West-Nielsen, Funktionsleder, Klimadatastyrelsen

Morten Lindegaard, Chefkonsulent, Klimadatastyrelsen

Lars Erik Storgaard, Chefkonsulent, Klimadatastyrelsen

FDP-sekretariatet
Nicklas Vandfeldt Hansen, Klimadatastyrelsen

Version af leverance: 1.0

Leverancebeskrivelse fremgår på **Forsyningsdigitaliseringsprogram.dk**

Indholdsfortegnelse

1. Indledning og formål

1.1 Baggrund

Retningslinjerne for dataformater for deling af forbrugsdata er udarbejdet af udvalget Tværgående Arkitektur og Udvikling (TAU) i regi af Forsyningsdigitaliseringsprogrammet (FDP). Retningslinjerne tager afsæt i målsætning 2 om ensretning af data og digitalisering, som er fastsat af Forum for Forsyningsdata (FFD).

FDP skal skabe et fælles grundlag for digitalisering i forsyningssektoren. Derfor er der igangsat et arbejdsspor vedr. interoperabilitet i regi af TAU, hvor relevante repræsentanter fra forsyningssektoren og myndigheder er repræsenteret. Arbejdssporet arbejder blandt andet med leverance 3 om fælles retningslinjer for dataformater og metadata i Forsyningsdigitaliseringsprogrammet.

Retningslinjerne udgør en del af TAU leverance 3 ”Fælles retningslinjer for dataformater og metadata i partnerskabet for forbrugs- og produktionsdata”, i arbejdsprogrammet for TAU under FDP.

Retningslinjerne gælder for dataformater for deling af forbrugsdata.

TAU anbefaler, at forsyningssektoren udveksler forbrugsdata i dataformaterne CSV, JSON, XML og Apache Parque, og at retningslinjerne får status som en fælles referenceramme for anvendelse af dataformater til deling af forbrugsdata med anvendere, som forsyningssektoren i relevant omfang følger. Implementeringen af retningslinjerne anbefales at ske løbende efterhånden som systemer opdateres eller fornyes.

Flere oplysninger om Forsyningsdigitaliseringsprogrammet findes på: www.forsyningsdigitaliseringsprogram.dk

1.2. Formål

Formålet med retningslinjerne er at beskrive, hvilke dataformater forsyningssektoren bør anvende, når forbrugsdata deles med anvendere. Retningslinjerne understøtter en mere ensartet brug af dataformater, så det bliver lettere at sammenstille, dele og genanvende data i forsyningssektoren, andre sektorer og hos anvendere.

Retningslinjer beskriver overordnet brugen af CSV, JSON, XML og Apache Parquet. Valget af dataformat er nødvendigt som første skridt mod interoperabilitet, men er ikke i sig selv tilstrækkeligt. De næste skridt omfatter den konkrete datamodellering, som er uden for disse retningslinjers scope. Læs mere om dataformater og datamodeller i afsnittet med yderligere information om dataformater.

Retningslinjerne dækker ikke GIS-specifikke geodataformater, men fokuserer på formater til deling af forbrugsdata med anvendere. Geografiske oplysninger (fx koordinater) kan dog stadig skrives standardiseret som tekst i de generelle formater.

Retningslinjerne henvender sig primært til IT-arkitekter, der designer IT-løsninger til deling af data med anvendere i forsyningssektoren, men det er tilstræbt, at retningslinjerne også kan læses af andre.

2. Anbefaling til implementering

TAU anbefaler, at de udarbejdede retningslinjer får status som en fælles referenceramme for anvendelse af dataformater til deling af data med anvendere, som forsyningssektoren i relevant omfang følger. Såfremt retningslinjerne ikke anvendes, bør det overvejes, hvorfor der afviges fra retningslinjerne.

Implementeringen af retningslinjerne anbefales desuden at ske løbende, efterhånden som systemer

opdateres eller fornyes, og dermed ikke med en hård implementeringsdeadline. Anbefalinger til anvendelse af formaterne er beskrevet i afsnit om valg af dataformater og systemunderstøttelse.

TAU anbefaler, at de udarbejdede retningslinjer revideres årligt, således at de løbende kan blive opdateret ud fra erfaringer med dem i forsyningssektoren og den teknologiske udvikling. Dette vil i regi af Forsyningsdigitaliseringsprogrammet ske i TAU via TAU-sekretariatet.

FDP-sekretariatet har udarbejdet en overordnet plan for indsatsområder, der skal understøtte udbredelse og anvendelse af de udarbejdet samt kommende retningslinjer og standarder i Forsyningsdigtaliseringsprogrammet.

Som led i implementering af retningslinjerne for dataformater for deling af forbrugsdata igangsættes indsatser for at understøtte en bred udbredelse anvendelse. Dette indebærer dialog med brancheorganisationer og centrale aktører om udbredelse og anvendelse. Derudover en indsats rettet mod anvendere og IT-leverandører, bl.a. høring for at sikre forankring, validering og praktisk anvendelighed. Endelig følges der løbende op på graden af anvendelsen af retningslinjerne for at skabe viden om anvendelsen samt understøtte deres løbende tilpasning jf. Plan for udbredelse og anvendelse af retningslinjer og standarder i FDP.

3. Governance og forankring

2.1 Governance i regi af Forsyningsdigitaliseringsprogrammet

I regi af FDP fungerer TAU som rådgivende organ, og kan gennemføre review på fx design og kvalitetssikring af datamodeller i forsyningssektoren i henhold til de udarbejdede retningslinjer.

2.2 Forankring i sektoren

Brancheorganisationerne vil bidrage til at understøtte udbredelse og forankring af leverancen via formidling af retningslinjerne i deres respektive sektorer gennem egne kanaler, fx branchestandarder, hjemmesider, nyhedsbreve og konferencer mv.

3. Anbefalet anvendelse af retningslinjerne

Retningslinjerne fungerer som en ramme for valg af dataformater til deling af forbrugsdata. Retningslinjerne udgør en liste af overordnede dataformater og viser, i hvilke forskellige brugssituationer, hvert format er egnet.

Retningslinjerne må ikke forveksles med en fælles datamodel for forsyningssektorens data. Det vil være en datamodel, som fastlægger, hvordan data struktureres. En datamodel fastlægger, hvilke oplysninger, der skal være i data dvs., hvordan felter/kolonner/attributter navngives i data, hvilke datatyper de har, og om de er obligatoriske eller valgfrie. Retningslinjerne kan konkret anvendes til valg af dataformater til datadelingsløsninger i forsyningssektoren.

Retningslinjerne tager udgangspunkt i, at valg af dataformater til deling af data med anvendere inden for forsyningssektoren ikke skal være anderledes end i andre sektorer, og skal sikre, at valg af dataformater til deling af data med anvendere i forsyningssektoren følger samme praksis.

4. Værdiskabelse

Anvendere af forbrugsdata efterspørger, at data leveres i standardiserede formater som CSV, JSON eller XML, jf. Anvenderønsker til forbedret forsyningsdataadgang. Både CSV, JSON og XML er med i retningslinjerne. CSV er et udbredt filformat, der er let at bruge i forskellige sammenhænge. JSON er ligeledes et udbredt format, der understøttes bredt i programbiblioteker og anvendes i moderne web-API'er. Endelige er XML et udbredt filformat med en bred understøttelse og gode muligheder for at validere strukturen på data.

Ved at følge retningslinjer og anvende standardiserede formater kan data lettere deles med anvendere og udveksles på tværs af systemer. Resultatet er reduceret ressourcetræk og hurtigere eksekvering i både drift

og udvikling af løsninger og services til deling af forbrugsdata med anvendere.

5. Retningslinjer for dataformater

Arbejdssporet anbefaler at forbrugsdata udveksles i dataformaterne CSV, JSON, XML og Apache Parquet. De fire formater er åbne standarder, som allerede benyttes bredt i sektoren forvejen og er bredt understøttet i værktøjer og programbiblioteker til fx Java og Python. Selvom JSON i mange sammenhænge kan benyttes som et enklere alternativ til XML, så er XML taget med i retningslinjerne, da det stadig har stor udbredelse, fx i vandsektoren. Følgende afsnit beskriver anvendelsesområdet for hvert af de nævnte formater, centrale kendetegn og henvisning til specifikationer samt fordele, ulemper og eksempler.

5.1 CSV

Anvendelsesområde

Brug CSV til løsninger, hvor anvenderen downloader filer med data.

Beskrivelse

CSV-filer er et tekstbaseret, menneskeligt læsbart (human-readable) dataformat, der er velegnet til data i tabeller. Data er struktureret i kolonner, og der er en linje for hver data record. CSV (Comma-Separated Values) er i daglig tale en kommasepareret fil, dvs. en tekstfil, hvor dataværdier i kolonner er adskilt med kommategn. Der benyttes dog ofte andre tegn end komma, fx semikolon, til at adskille kolonner.

Specifikation

En specifikation af CSV-filer er lagt frem i RFC 4180, men der er en bred praksis for CSV-filer, der ofte ikke følger denne specifikation fuldt ud. RFC 4180 er et dokument under The Internet Engineering Task Force (IETF®), der producerer tekniske dokumenter med henblik på at forbedre internettet.

Fordele og ulemper

Fordele ved CSV er, at det er menneskeligt læsbart. Filer i dette format kan importeres i almindelige regnearksprogrammer, fx Microsoft Excel og LibreOffice Calc.

Ulemper er, at strukturen af data i en CSV-fil er begrænset til den, der svarer til en enkelt tabel. Desuden har formatet ikke indbyggede datatyper eller formalismer til at fastlægge og håndhæve datatyper og

Eksempel
Eksempel på udsnit af CSV-fil fra Energi Data Service:

```
HourUTC;HourDK;ConnectedArea;SharePPM;ShareMWh
2025-03-05 06:00:00;2025-03-05 07:00:00;DK1;692681;5001,428025
2025-03-05 06:00:00;2025-03-05 07:00:00;DK2;274509;1982,062180
2025-03-05 06:00:00;2025-03-05 07:00:00;NO2;0;0,000000
2025-03-05 06:00:00;2025-03-05 07:00:00;SE;32809;236,895858
2025-03-05 06:00:00;2025-03-05 07:00:00;GE;0;0,000000
2025-03-05 06:00:00;2025-03-05 07:00:00;NL;0;0,000000
2025-03-05 05:00:00;2025-03-05 06:00:00;DK2;262692;1806,697900
2025-03-05 05:00:00;2025-03-05 06:00:00;DK1;737307;5070,907512
```

Første linje i filen indeholder navne på kolonnerne. Herefter indeholder hver linje en data record med værdier adskilt med semikolon. Indholdet af kolonnerne er beskrevet i metadata, der ikke er en del af CSV-filen.

5.2 JSON

Anvendelsesområde
Brug JSON til løsninger, hvor anvenderen henter data via et web-API. Brug evt. JSON i løsninger, hvor anvenderen downloader filer med data.

Beskrivelse
JSON (JavaScript Object Notation) er et tekstbaseret, menneskeligt læsbart dataudvekslingsformat. Der er en bred understøttelse med forskellige programbiblioteker, og som navnet antyder, er det baseret på JavaScript-syntaks. JSON benyttes i dag i mange API'er og webudvikling

Specifikation
JSON er dokumenteret på json.org og i The JSON Data Interchange Standard.

Fordele og ulemper
Fordele ved JSON er, at det er menneskeligt læsbart, da det er tekstbaseret, og at det benyttes i mange moderne webteknologier. Yderligere fordele er, at en hierarkisk struktur kan repræsenteres i JSON-filer, og struktur med indbyggede datatyper og lister kan beskrives med JSON Schema.

En ulempe er, at JSON ikke et kompakt format, da det bygger på samlinger af navn/værdi-par og dermed typisk indeholder mange gentagne navne på værdier.

Eksempel
Eksempel på JSON-svar fra API hos Energi Data Service:


```
{
  "total": 1095743,
  "limit": 2,
  "dataset": "ConsumptionCoverageNationalDecl",
  "records": [
    {
      "HourUTC": "2025-03-05T06:00:00",
      "HourDK": "2025-03-05T07:00:00",
      "ConnectedArea": "DK1",
      "SharePPM": 692681,
      "ShareMWh": 5001.428025
    },
    {
      "HourUTC": "2025-03-05T06:00:00",
      "HourDK": "2025-03-05T07:00:00",
      "ConnectedArea": "DK2",
      "SharePPM": 274509,
      "ShareMWh": 1982.062180
    }
  ]
}
```


Eksemplet er svar på en forespørgsel på to data records fra datasættet Consumption Coverage, Hourly National Statistics. Svaret indeholder først oplysninger relateret til brugen af API'et, og selve data er i en liste med data records under "records". Hvis der sammenlignes med eksemplet på CSV, ses det, at flere data records vil fylde mere i JSON, da navnene på datafelterne gentages for hver data record. JSON-eksemplet indeholder kun de to første records fra CSV-eksemplet, men fylder mere end de tre øverste linjer i CSV-eksemplet. JSON-eksemplet er formateret af hensyn til læsbarhed.

5.3 XML

Anvendelsesområde
Brug XML til løsninger, hvor anvenderen henter data via et web-API. Brug evt. XML i løsninger, hvor

anvenderen downloader filer med data. XML er især velegnet i løsninger, hvor der er behov for en standardiseret tilgang til validering af struktur på data.

Beskrivelse
XML (eXtensible Markup Language) er et opmærkningssprog, der kan benyttes til at strukturere data i tekstfiler. XML er menneskeligt læsbart, men er også designet til at blive processeret automatisk af software, og der er en bred understøttelse med forskellige programbiblioteker og andre værktøjer. XML er et generelt sprog, der ikke på forhånd dikterer, hvordan data skal struktureres.

Specifikation
Der er information om XML på https://www.w3.org/XML/, og en specifikation findes på https://www.w3.org/TR/REC-xml/.

Fordele og ulemper
Fordele ved XML er, at det er menneskeligt læsbart, da det er tekstbaseret, og at det er udbredt og velkendt i mange sammenhænge. Der er en fordel, at en hierarkisk struktur kan repræsenteres i XML, og at struktur med indbyggede datatyper mv. kan beskrives med fx XML Schema Definition (XSD), der kan benyttes til validering af strukturen.

En ulempe er, at XML ikke er et kompakt format, da det benytter sammenhængende start- og slut-tags, hvorved der typisk kommer mange gentagne navne på værdier.

Eksempel
Eksempel på data i et XML-dokument:

```
<?xml version="1.0" encoding="UTF-8"?>
<fme:xml-tables xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
                xmlns:fme=http://www.safe.com/xml/xmltables
                xsi:schemaLocation="http://www.safe.com/xml/xmltables
ConsumptionCoverageNatonalDecl.xsd">
          <fme:ConsumptionCoverageNationalDecl-table>
                    <fme:ConsumptionCoverageNationalDecl>
                              <fme:HourUTC>20250305060000</fme:HourUTC>
                              <fme:HourDK>20250305070000</fme:HourDK>
                              <fme:ConnectedArea>DK1</fme:ConnectedArea>
                              <fme:SharePPM>692681</fme:SharePPM>
                              <fme:ShareMWh>5001.428025</fme:ShareMWh>
                    </fme:ConsumptionCoverageNationalDecl>
                    <fme:ConsumptionCoverageNationalDecl>
                              <fme:HourUTC>20250305060000</fme:HourUTC>
                              <fme:HourDK>20250305070000</fme:HourDK>
                              <fme:ConnectedArea>DK2</fme:ConnectedArea>
                              <fme:SharePPM>274509</fme:SharePPM>
                              <fme:ShareMWh>1982.06218</fme:ShareMWh>
          </fme:ConsumptionCoverageNationalDecl>
          </fme:ConsumptionCoverageNationalDecl-table>
</fme:xml-tables>
```

Eksemplet indeholder to data records fra datasættet Consumption Coverage, Hourly National Statistics, der også er benyttet i de tidligere eksempler på CSV og JSON. Eksemplet er udarbejdet ved at læse data fra CSV med ETL-værktøjet FME og skrive data i et XML-dokument. Derudover er XML-eksemplet formateret af

hensyn til læsbarhed. Hvis man sammenligner med CSV-eksemplet, ses det, at data fylder mere, når det er skrevet i XML, og derfor er kun to data records taget med i eksemplet. Man kunne også have repræsenteret JSON-eksemplet i XML, da XML har rig mulighed for at indeholde hierarkiske strukturer i data.

5.4 Apache Parquet

Anvendelsesområde

Brug Apache Parquet til løsninger, hvor anvenderen downloader filer med data, og der er tale om meget store datamængder. Det afhænger af konteksten, hvad der er meget store datamængder. Både størrelserne på enkelte filer, den samlede mængde data i løsningen og overførselshastigheder mv. kan indgå i valg af Apache Parquet.

Beskrivelse

Apache Parquet er et binært, ikke-menneskeligt-læsbart filformat, dvs. en Parquet-fil er ikke blot en tekstfil og data giver derfor ikke mening i en teksteditor. Parquet er designet til at lagre data effektivt med komprimering, således at filerne fylder mindre end rå data i tekst. Parquet understøttes med program-biblioteker i en række forskellige programmeringssprog og værktøjer – det er et open source, kolonneorienteret filformat, der indgår i Apache Hadoop-økosystemet, som er en samling af forskellige softwareværktøjer.

Specifikation

Parquet er dokumenteret på parquet.apache.org. Den seneste version af Apache Parquet-formatet er ved udarbejdelse af disse retningslinjer version 2.11.0 fra marts 2025. Selve Parquet-formatet er beskrevet i parquet-format, og en Java-implementering med programbiblioteker til at læse og skrive i formatet findes i parquet-java. Begge dele er med i Apache Parquet-projektet. Det anbefales at benytte de nyeste versioner af hensyn til fejlrettelser.

Fordele og ulemper

Det er en fordel ved Apache Parquet, at det tilbyder en meget kompakt repræsentation af data med blandt andet indbyggede datatyper og komprimering. Da Parquet er et kolonneorienteret format, kan databehandling optimeres ved at benytte information i filen til kun at tilgå relevante kolonner, således at det ikke er nødvendigt at læse hele filen for at hente relevante kolonner ud. Ulemperne ved Apache Parquet er, at det er mere komplekst end tekstbaserede formater, og at det ikke er menneskeligt læsbart, hvilket stiller større krav til værktøjer, understøttelse med software og viden om formatet.

Eksempel

Da Apache Parquet er et binært format, kan der ikke gives meningsfyldte eksempler på samme måde som for CSV, JSON og XML. De første fire bytes og de sidste fire bytes i en Parquet-fil skal være ”PAR1”, men derudover er filen ikke menneskelig læsbar.

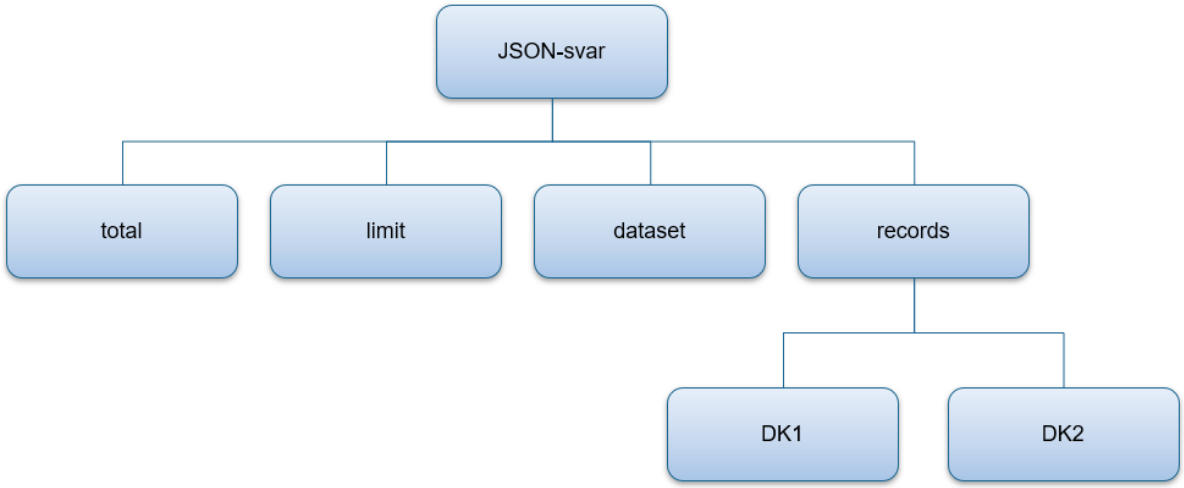
Forskellige ETL-værktøjer kan benyttes til at læse og skrive data i Parquet. Hvis man med et ETL-værktøjer læser en Parquet-fil på fx 60 MB og skriver data til en JSON-fil, så kan JSON-filen fylde ca. 2 GB.

6. Om valg af dataformater

Valg af dataformater til en konkret IT-løsning bør foretages ud fra en konkret vurdering, der tager hensyn til blandt andet konteksten for brug af formatet, strukturer i data og ikke mindst datamængder. Ved udvikling af API'er, vil det fx være naturligt at vælge JSON, mens CSV kan være et godt valg til download af små datamængder fra en hjemmeside, fx en måneds forbrug for en husstand. Evt. komprimering af filer bør også indgå i overvejelserne, især ved en download-løsning med statiske filer. Der kan fx opnås en fin pladsbesparelse ved at komprimere JSON- og XML-filer til ZIP-filer. Hvis en download-løsning skal håndtere

meget store datamængder, fx mange store datasæt i statiske filer til download fra en hjemmeside og løbende tilføjelse af nye filer til download, så kan Parquet være et hensigtsmæssigt valg til en kompakt lagring og udstilling af data. Ved valg af dataformater kan der i en løsning til download af filer udarbejdes realistiske prøvedata i forskellige formater med og uden kompression for at vurdere, hvad der er bedst egnet til data i den pågældende struktur og opdeling (fx geografisk og tidsmæssig opdeling).

I modsætning til CSV og Apache Parquet, så er JSON og XML velegnet til hierarkiske data, dvs. data i JSON og XML kan have en hierarkisk struktur i stedet for blot en tabel. Derudover kan flere tabeller indgå i samme JSON- eller XML-fil. Når JSON eller XML benyttes til svar i et API, vil svaret typisk indeholde både selve data og statusinformation pakket i samme svar, jf. tidligere eksempel på JSON. Figur 1 illustrerer den hierarkiske struktur i JSON-svaret. Statusinformationen indeholder i eksemplet felterne ”total” med antallet af data records i de fremsøgte data, ”limit” med antallet af data records i svaret (givet ved en parameter i forespørgslen til API), og ”dataset” med navnet på datasættet. Selve data er i eksemplet indeholdt i listen ”records” med to data records for områderne DK1 og DK2.



Figur 1: Struktur i JSON-svar

De enkelte felter for statusinformation og listen med data records er vist i figuren. Felterne i de to data records, der illustreres med DK1 og DK2, er udeladt af hensyn til overskuelighed.

Man har benyttet XML i webtjenester i mange år, men moderne web-API benytter i dag ofte JSON.

7. Systemunderstøttelse

CSV, JSON, XML og Apache Parquet er formater, der kan håndteres af moderne ETL-værktøjer, og de kan derfor benyttes, når der konstrueres et system eller en datapipeline med gængse rammeværk og software-produkter. Derudover er der programbiblioteker i programmeringssprog, der kan benyttes til at læse og skrive data i formaterne. Ved udvikling af softwaresystemer til udveksling af data med anvenderne, bør de anbefalede dataformater betragtes, når systemet kravsættes, designes og udvikles.

Gængse regnearksprogrammer som Microsoft Excel og LibreOffice Calc kan importere CSV-filer, således at anvenderen vil kunne importere og arbejde med data i regneark, hvis anvendelsen ikke nødvendiggør implementering eller konfigurering af et softwaresystem. Gængse regnearksprogrammer har som regel en øvre grænse for, hvor mange rækker det kan indeholde, mens en CSV-fil ikke har den samme begrænsning.

Da CSV-filer uden videre kan importeres i almindelige regnearksprogrammer, er formatet et enkelt alternativ, når data ellers ville blive delt i Excel.

Selvom JSON (JavaScript Object Notation) er baseret på en del af standarden for programmeringssproget JavaScript, så må det betragtes som et uafhængigt, selvstændigt dataformat. På json.org er der links til en lang række programbiblioteker, der understøtter JSON, i udbredte programmeringssprog som fx C, C++, C#, Java, JavaScript, Python og Rust. JSON er ofte det foretrukne dataformat i moderne API'er, og derfor er der også understøttelse af JSON i rammeværker til API-udvikling.

Der er en bred understøttelse af XML, herunder programbiblioteker i programmeringssprog som fx Java og Python samt rammeværket .NET, som understøtter C#. Mange internationale standarder er baseret på XML, og der er en lang række af produkter og teknologier baseret på XML. Som tidligere nævnt, kan struktur fx beskrives i XML Schema, og egenskaber for data i XML-dokumenter kan yderligere beskrives med regler og valideres med Schematron. Dvs. der er systemunderstøttelse for validering af struktur som "skemavalidering" med XSD samt validering af indhold og sammenhænge i data ud fra regler med Schematron. XSLT (eXtensible Stylesheet Language Transformations) kan benyttes til at transformere XML-dokumenter og benyttes ofte i forbindelse med Schematron.

I regi af Apache Software Foundation® har Apache Parquet underprojekter, der leverer programbiblioteker til programmeringssprogene Java, C++, Rust og Golang. Derudover understøttes Parquet af projekter som Apache Arrow, og det indgår fx som underliggende filformat i rammeværker som Delta Lake, der igen benyttes af kommercielle analyseplatforme som Azure Databricks.

8. Ordliste

API (Application Programming Interface):
Et API er en snitflade til software, der kan benyttes ved udvikling af softwareapplikationer. I dag benyttes API ofte synonymt med web-API, således at det betegner en snitflade, der kan tilgås over internettet, fx med HTTP-protokollen.

ETL (Extract, Transform og Load):
ETL er en proces (udtræk, transformation og indlæsning), hvor data hentes fra en eller flere datakilder, transformeres og evt. sammensættes for at blive læst ind i et datalager eller bestemt dataformat.

Data record:
En samling af dataværdier, der hører sammen. Dataværdierne kan være af forskellige typer. En data record kan fx bestå af et tidsstempel, identifikationsnummer på en måler og en målt værdi. En data record er ofte repræsenteret som en række i en databasetabel.

Human-readable:
Når et format er human-readable, kan data læses af et menneske. Fx kan dataværdier og navngivning af datafelter være i klar tekst, så et menneske kan læse dem direkte i en teksteditor.

JavaScript:
Programmeringssprog designet til internettet. JavaScript kører i webbrowsere og benyttes til at implementere funktionalitet på hjemmesider.

UML:
Unified Modeling Language (forenet modelleringssprog). Det er et standardiseret, generelt modelleringssprog, der bruges til at visualisere, specificere, konstruere og dokumentere artefakter af

softwareintensive systemer. Selvom det primært bruges i softwareudvikling, kan det også anvendes til at modellere forretningsprocesser og andre ikke-softwaresystemer.

9. Yderligere information om dataformater

Retningslinjerne i dette dokument omhandler dataformaterne CSV, JSON, XML og Apache Parquet på et overordnet niveau uden at specificere indhold i data. Valg af overordnede dataformater er en nødvendig, men ikke tilstrækkelig, forudsætning for interoperabilitet. Der bør også betragtes struktur i data og de bagvedliggende datamodeller, der kan understøttes på forskellig vis i dataformater.

9.1 Struktur i data

Data kan findes i tabelform, organiseret i rækker og kolonner. Forbrugsdata er i dag forholdsvis simple og passer ofte ind i en tabel. Eksemplet på data i en CSV-fil i tidligere afsnit illustrerer, hvordan data i dag findes i tabelform.

Data kan indeholde hierarkiske strukturer. I så fald er JSON et velegnet dataformat, da det oprindeligt er designet til at indeholde objekt-hierarkier. XML ligeledes et velegnet dataformat til træstrukturer og hierarkier af elementer.

9.2 Datatyper i dataformater

Datatyper spiller en vigtig rolle i specifikation af datastrukturer, datamodellering og håndtering af data. Dataformater kan have indbyggede datatyper, der blandt andet gør det lettere at validere data.

Datatyper kan generelt være primitive datatyper eller sammensatte datatyper. Primitive datatyper kan fx være heltal, decimaltal, sandhedsværdier (true og false) og tegn. Sammensatte datatyper kan konstrueres ud fra andre datatyper. Fx kan sammensatte datatyper være "lister af heltal" eller "par bestående af et heltal og en sandhedsværdi".

Vedbenyttelse af CSV som udvekslingsformat, er der ikke indbyggede datatyper, og det skal særskilt beskrives over for anvenderne, hvordan tekst i de forskellige kolonner skal fortolkes. Nogle ETL-værktøjer kan dog aflede en datatype ud fra teksten i en kolonne, men den afledning er ikke altid korrekt.

JSON-dokumenter dannes af to strukturer: Objekt, som er en samling af par med navn og værdi, og ordnet liste af værdier. I JSON kan værdier være objekter, arrays, tal, tekststrenge samt "true", "false" eller "null". De primitive datatyper er således tal, tekststrenge, sandhedsværdier (boolean) og null. Når værdier kan være objekter, er der mulighed for at repræsentere hierarkiske strukturer i data som objekter, der indeholder andre objekter. (For en fuldstændig gennemgang henvises til json.org.)

Når XML-dokumenter beskrives med XML Schema (XSD), så kan der benyttes en lang række datatyper som er specificeret i W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes. Der er primitive datatyper til tekststrenge, sandhedsværdier, tal og tidsangivelser: string, boolean, decimal, float, double, duration, dateTime, time, date etc. Og der er en lang række andre indbyggede datatyper, der er afledt af de primitive datatyper, fx normalizedString, integer, nonPositiveInteger og dateTimeStamp. Derudover er der rig mulighed for at definere datatyper og beskrive hierarkiske strukturer i data.

Apache Parquet indeholder forskellige primitive datatyper til sandhedsværdier, tal og tekststrenge: Boolean, Int32, Int64, Int96, Float, Double, Byte_array og Fixed_len_byte_array. Dette svarer til forskellige datatyper, som kan forventes i programmeringssprog. Derudover har Parquet logiske datatyper, der bygger oven på de primitive datatyper, dvs. de logiske datatyper (Logical Types) udvider de primitive datatyper ved

at specificere, hvorledes de skal fortolkes. Oplysninger om typerne for de forskellige kolonner er indlejret metadata i Parquet-formatet. (For en fuldstændig gennemgang henvises til parquet.apache.org.)

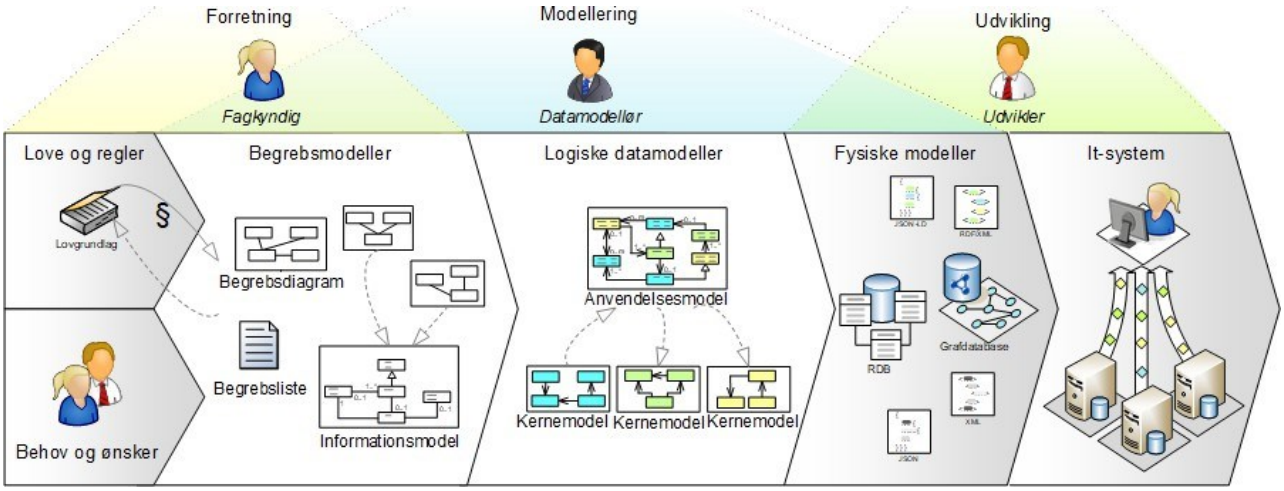
10. Dataformater og datamodeller

Datamodeller beskriver data og især struktur i data med relationer mellem forskellige entiteter. Forskellige dataformater kan benyttes til data, der er beskrevet af den samme model. Dog skal der tages praktiske hensyn til strukturen i datamodellen. Dvs. nogle formater kan indeholde flere entiteter og hierarkiske strukturer i samme fil, mens andre kun indeholder data for én entitet i en fil. Hvis relationelle databaser benyttes, vil hierarkiske strukturer i en datamodel typisk blive implementeret med flere databasetabeller, og allerede her vil der være foretaget en transformation fra den logiske model til en fysisk model med flade tabeller, der kan repræsenteres i de forskellige dataformater behandlet i disse retningslinjer.

Ud fra en datamodel, kan struktur på data blandt andet beskrives med JSON Schema, hvormed strukturen af data fastlægges med datatyper og evt. hierarki. Herefter er det muligt at validere, at konkrete data i JSON lever op til skemaet. Tilsvarende kan XML Schema benyttes til at validere struktur på data i XML.

Dataformater er første skridt, og datamodellering kombineret med teknologier som JSON Schema og XML Schema, der kan benyttes til at validere data, er de næste skridt på vejen til interoperabilitet og bedre udveksling af data i forsyningssektoren med anvendere.

I de følgende afsnit beskrives kort, hvilke forskellige former for modeller, der kan udarbejdes i løbet af en datamodelleringsproces som illustreret i figur 2.



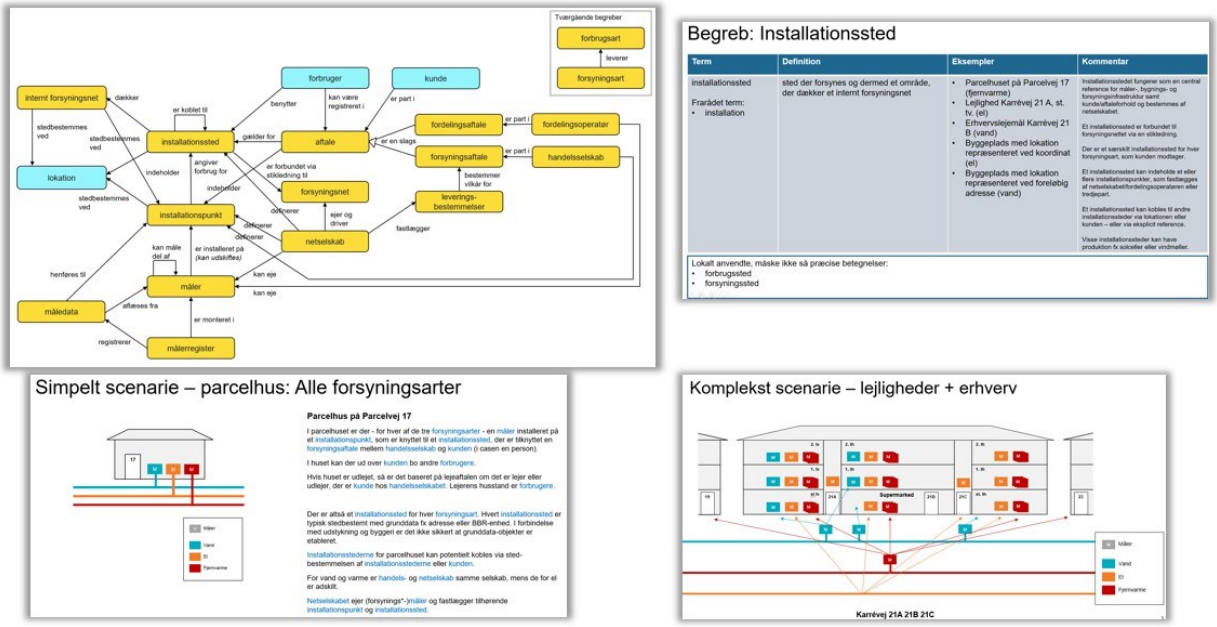
Figur 2: Forskellige former for modeller i en udviklingsproces. Kilde: https://arkitektur.digst.dk/metoder/begrebs-og-datametoder/regler-begrebs-og-datamodellering/de-faellesoffentlige-regler-begrebs

Udformningen af konkrete datamodeller for forsyningssektoren og en detaljeret indføring i datamodellering ligger uden for rammerne af disse retningslinjer, og der henvises til Fællesoffentlig Digital Arkitektur (FDA) for vejledning, regler og metoder for begrebs- og datamodellering.

Begrebsmodeller

Begrebsafklaring er en vigtig det af datamodellering. I regi af FDP er der udarbejdet en overordnet begrebsmodel, der skal bidrage til en fælles forståelse og et ensartet sprog på tværs af forsyningssektoren. Begrebsmodellen kan danne udgangspunkt for udarbejdelse af mere detaljerede delmodeller efter behov.

Begrebsmodellen omfatter et begrebsdiagram med vigtige begreber og deres indbyrdes relationer samt detaljerede beskrivelser af begreberne med term, definition, eksempler og kommentarer. Derudover beskrives en række scenarier for forsyning af el, fjernvarme og vand til forskellige typer af forbrugs-lokationer for at trykprøve begreberne. Figur 3 illustrerer begrebsmodellen og tilhørende materiale.



Figur 3: Materiale fra begrebsmodellen

Der henvises til Fælles retningslinjer for begreber og deres relationer (TAU leverance 2) for en egentlig gengivelse af de identificerede fælles begreber. Begrebsmodellen blev udarbejdet med metoder fra Fællesoffentlig Digital Arkitektur (FDA).

10.1 Logiske datamodeller

Med udgangspunkt i begrebsafklaring og begrebsmodeller, kan der udarbejdes logiske datamodeller med flere detaljer om de entiteter, der indgår. Dvs. de logiske modeller indeholder information om, hvilke attributter de forskellige entiteter har.

Logiske datamodeller udarbejdes ofte i UML eller lignende sprog med en grafisk notation. Hvis man benytter UML, vil man typisk udarbejde et klassediagram, hvor hver entitet er repræsenteret med en klasse (af objekter), og relationer mellem entiteterne/klasserne også er beskrevet. Arbejdet med logiske modeller i fx UML er gerne værktøjsunderstøttet med specialiseret software.

10.2 Fysiske datamodeller

Ud fra de logiske datamodeller, kan man udarbejde fysiske datamodeller, der typisk omfatter definitioner af databaseskemaer og databasetabeller. Denne proces kan være værktøjsunderstøttet, således at man automatisk kan danne definitioner af databasetabeller ud fra en UML-model, men kan også foretages som en systematisk, manuel proces.

Fysiske datamodeller omfatter også skemaer, fx JSON Schema og XML Schema, som fastlægger struktur og indhold i dataformater. Når de fysiske datamodeller til databaser og dataformater dannes ud fra samme logiske model kan man sikre en god overensstemmelse og gøre det lettere at udstille data.