SDFI

ARBEJDSGRUPPE: DATA OG INDLÆSNING

Moderniseret Datafordeler

Dato: 6. Oktober 2023
Version: 1.0
Forfatter: Marcus Quistgaard og René Ravn
Kontakt: mwq@netcompany.com, rra@netcompany.com

netcompany

Arbejdsgruppe: Data og Indlæsning

- MT5: Nutidsdatabase med aktuelle data
- MT9: Optimering af løbende indlæsning af data
- MT10: Optimering af totalindlæsning og datamodelændringer

Præsentationsrunde

- Hvem er I, og hvad er jeres relation til Datafordeleren?
  - Er I Anvendere, Registre, noget helt tredje?

Agenda

- Nutidsdata
  - Bitemporale opslag i Datafordeleren i dag
  - Vision om ”Nutids”-data
- Indlæsning
  - Indlæsning i dag
  - Moderne indlæsning
    - Dialog omkring problemer og ønsker
    - Vision om moderne indlæsnings flow
    - Håndtering af parallelisering af indlæsning
- Totalindlæsning
  - Totalindlæsning i dag
  - Vision for optimeret totalindlæsning
  - Datamodel versionering

Formål med dagens workshops

- Formålet i dag er at
  - Indsamle behov og ønsker til Datafordelerens fremtidige funktionalitet med afsæt i produktvisionen
  - Informere om NCs vision omkring de berørte emner.

- Formålet i dag er **ikke** at
  - Have et endeligt design
  - Træffe endelige beslutninger om en konkret løsning
  - Gå i tekniske detaljer omkring implementation af en konkret løsning

TID I DATA

Bitemporalitet i dag – Gruppe dialog

- Gå sammen i grupper af 3-4 personer
- Tal sammen om følgende punkter i 10-15 minutter (punkterne er på næste slide)
- Hver gruppe præsenterer hovedemner fra deres dialog i plenum
- Fælles dialog

Bitemporalitet i dag – Snak sammen i grupper (10-15 minutter)

- Hvordan arbejder I med (bi)temporal data i jeres system?
  - *Eller sagt på en anden måde:*
    Gør I brug af både registreringstid-dimensionen og virkningstid-dimensionen?
    Og sætter I altid disse dimensioner til samme point-in-time, eller sætter I dem til forskellige tidspunkter?

- Hvilke tidsperioder gør I brug af på andre registre på DAF?
  - Nutid?
  - Point-in-time fremad/bagud eller periode-opslag?

- Hvordan modtager I de bitemporale data fra DAF, Filudtræk eller tjenester?

- Hvilke problemer har I med DAFs nuværende tjenester/Filudtræk, set fra et bitemporalitets perspektiv?

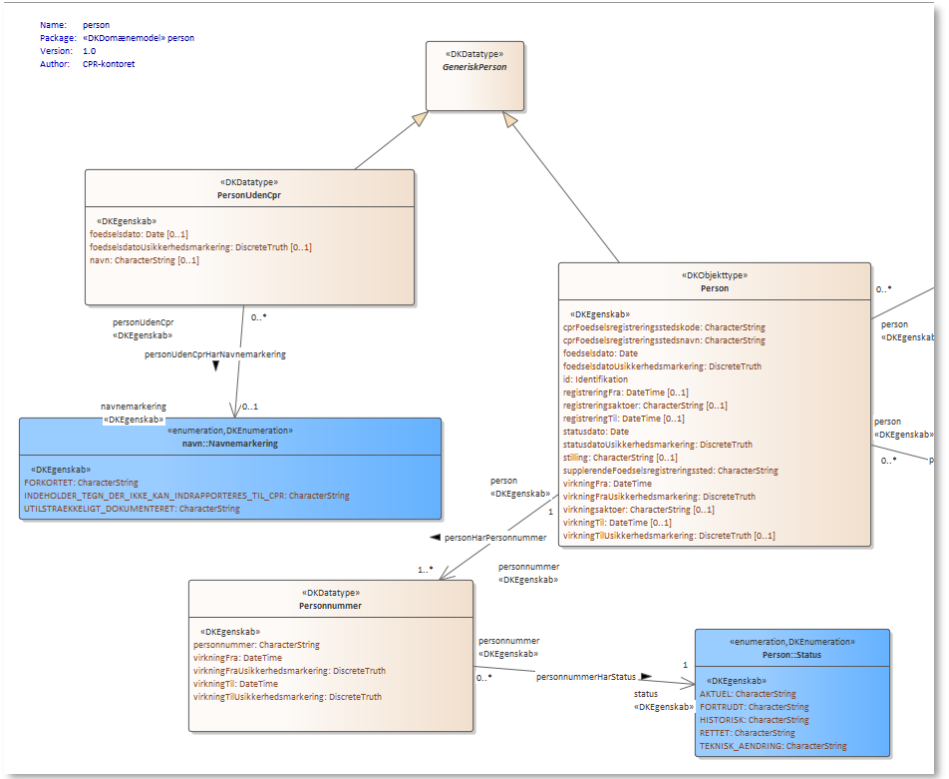
Bitemporale opslag i Datafordeleren i dag

- Muligheden for bitemporale opslag er begrænset af hvad registrene specificerer i deres DLS'er.
  - Opslag efter nutidsdata er generelt godt understøttet.
  - Opslag fremad og tilbage i tid er understøttet i nogle tjenester.
  - Opslag på tidsperioder (range-queries) er understøttet i nogle tjenester.
  - Bitemporale filtrering er ikke konsekvent på tværs af tjenester.
    - F.eks. laver tjenesten bitemporal filtrering på joinede tabeller?

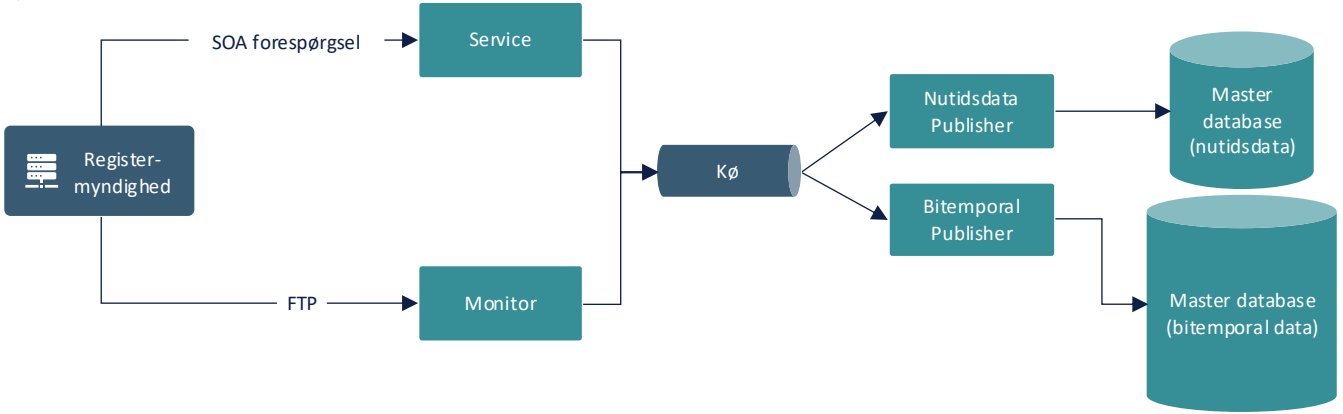
”NUTIDS”-DATA

Formålet med Nutidsdata

- Grunddatamodellen er i sig selv kompliceret. Tilføjelse af to tidsakser gør det ikke bedre.
- Modellen er modelleret uden tidsdimension.
- Langt mindre data at behandle og forstå.
- Fjerner register implementeringsdetaljer af bitemporalitet fra anvendere når der spørges på tværs af registre.
- Simplere tilgang til geotjenester i nytid.



Nutidsdata ud fra Bitemporal data



- High level indlæses ”nutid” ud fra opdateringerne og alt gøres tilgængeligt i bitemporal data.
- Understøtter muligheden for at lave bitemporale opslag.

Definition af ”nutid”

- Nutid i *registrering*-dimension: registreringFra **<** NOW **&& (**registreringTil **>** NOW **OR NULL)**
- Nutid i *virkning*-dimension: virkningFra **<** NOW **&& (**virkningTil **>** NOW **OR NULL)**

- Status og sagsnumre og lign. er en filtrering som anvender skal lave efterfølgende. Her vil relationerne være en-til-*.

Vision om fysisk separat nutidsdatabase

- Hvorfor etablere en nutidsdatabase?
  - Simplificeret udvikling af interne DAF komponenter. Alt bitemporal logik vil ligge i indlæsning af data til databasen.
  - Størrelsen af databasen vil være mindre end den nuværende totale model.
  - Data laves nutid én gang ved indlæsning.

- Udfordringer med en fysisk indlæsning
  - Data i ”bitemporal” og nutidsdatabasen er ude af sync i indlæsningen.
  - Data-fejl hvor der er overlappende perioder *skal* håndteres på indlæsnings-tidspunkt.
  - ”Genberegning” af nutid kan være kompliceret
  - Fremdateringer skal ”ligge i kø” og først indlæses når de bliver gældende
- Point-in-time kan ikke håndteres i nutids-dimensionerne.
  - Begrænser gevinst for anvendere som bruger andet end nutid

Simplere alternativ: Logisk point-in-time database ovenpå bitemp database

- Der oprettes point-in-time procedurer på samtlige bitemporale register-tabeller i begge tidsdimensioner.
  - Procedurer og indexes optimeret til point-in-time.
  - Point-in-time kan hentes i begge tidsdimensioner, både fremad og tilbage i tid.
  - Dette simplificerer snitfladen for interne services.

- Fordele ved point-in-time visning.
  - Data er altid i sync.
  - Fremdateringer håndteres ‘gratis’.
- Ulemper
  - Data skalering på det totale datamængder.
  - ”Nutids”-beregningen sker on-the-fly i stedet.
  - Historiske data kan være ukomplete og kan give uhensigtsmæssige svar.

DELTA INDLÆSNING I DAG

Hvordan virker den nuværende indlæsningen i dag (tabuler data)

1. Register uploader zip-fil til SFTP server eller sender XML til SOAP service.
2. DAF finder næste sekvensnummer som skal indlæses.
3. DAF udfører skema-validering på den indsendte register-xml.
4. DAF indlæser register-xml til masterload database.
5. DAF overfører indlæst data fra masterload til S5 masterpublish database
6. DAF overfører indlæst data fra masterload til S0 masterpublish database
7. DAF genererer hændelser i masterload database
8. Kvittering sendes til registret at indlæsningen er færdig
9. PULL hændelser er klar til at blive hentet.
10. Event Genereringen til push-abonnementer startes
11. PUSH-hændelser sendes til abonnenter

Indhold af replikeringspakker

- Pakken kan indeholde en eller flere xml filer.
- Hver fil indeholder en liste af objekt ændringer.

  Objekt ændringerne indlæses i rækkefølge, fra top til bund

  Indholdet af store filer bliver løbende committet efter X rækker (typisk 2000).
- Her er *ingen* mulighed for parallelisering af pakker i dag da rækkefølge er meget vigtig:

  Ændringer i fil #1, kan blive overskrevet af fil #2.

  Objekt-ændring #1 i fil #1, kan blive overskrevet af objekt-ændring #2 i samme fil.

```
<cmn:DatafordelerDataDelivery xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:cmn="http://data.gov.dk/daf" xmlns:dar="http://data.gov.dk/dar">
    <cmn:Header>
        <cmn:ReplicationChannelID>22</cmn:ReplicationChannelID>
        <cmn:SequenceNumber>3064327</cmn:SequenceNumber>
        <cmn:TimeOfCreation>2022-03-09T15:07:00.000000Z</cmn:TimeOfCreation>
        <cmn:CoversPeriodFrom xsi:nil="true"/>
        <cmn:CoversPeriodTo xsi:nil="true"/>
        <cmn:DeliveryType>Delta</cmn:DeliveryType>
    </cmn:Header>
    <cmn:Data>
        <cmn:DataEntity>
            <cmn:Action>Update</cmn:Action>
            <cmn:CreateEvent>NoEvent</cmn:CreateEvent>
            <dar:Adressepunkt>
                <dar:forretningshændelse>3</dar:forretningshændelse>
                <dar:forretningsområde>54.15.10.25</dar:forretningsområde>
                <dar:forretningsproces>0</dar:forretningsproces>
                <dar:id_namespace>http://data.gov.dk/dar/adressepunkt</dar:id_namespace>
                <dar:id_lokalId>709e0873-ab76-42f1-bca3-3b930eeed614</dar:id_lokalId>
                <dar:registreringFra>2019-01-28T10:10:40.579691Z</dar:registreringFra>
                <dar:registreringTil>2022-02-22T12:26:10.684194Z</dar:registreringTil>
                <dar:registreringsaktør>DAR</dar:registreringsaktør>
                <dar:status>8</dar:status>
                <dar:virkningFra>2019-01-28T10:10:40.579691Z</dar:virkningFra>
                <dar:virkningsaktør>AjourføreDarSystem</dar:virkningsaktør>
                <dar:oprindelse_kilde>Adressemyn</dar:oprindelse_kilde>
                <dar:oprindelse_nøjagtighedsklasse>A</dar:oprindelse_nøjagtighedsklasse>
                <dar:oprindelse_registrering>2019-01-28T10:10:40.579691Z
                </dar:oprindelse_registrering>
                <dar:oprindelse_tekniskStandard>UF</dar:oprindelse_tekniskStandard>
                <dar:position>POINT (565595.53119106265 6224031.6480632983)</dar:position>
            </dar:Adressepunkt>
        </cmn:DataEntity>
        <cmn:DataEntity>
            <cmn:Action>Create</cmn:Action>
            <cmn:CreateEvent>NoEvent</cmn:CreateEvent>
            <dar:Adressepunkt>
                <dar:forretningshændelse>3</dar:forretningshændelse>
                <dar:forretningsområde>54.15.10.25</dar:forretningsområde>
                <dar:forretningsproces>0</dar:forretningsproces>
```

Udvidet datakvalitetssikring

- I dag har Datafordeleren kun meget overfladisk datakvalitetssikring, hvor der kun kigges på registerets replikeringsskema.
  - Registeret er data-ansvarlig og derfor er det også den validering DAF som minimum kan lave.
  - Konsekvensen er, at fejlbehæftet data først bliver opdaget når en anvender indmelder at en tjeneste crasher eller returnerer for meget data.

- En mulighed er løbende at køre en bitemporal validering af registerets datasæt
  - Såfremt der findes fejl vil registrene blive informeret om at de har sendt fejlbehæftet data, og bedes rette det.

Dialog i plenum

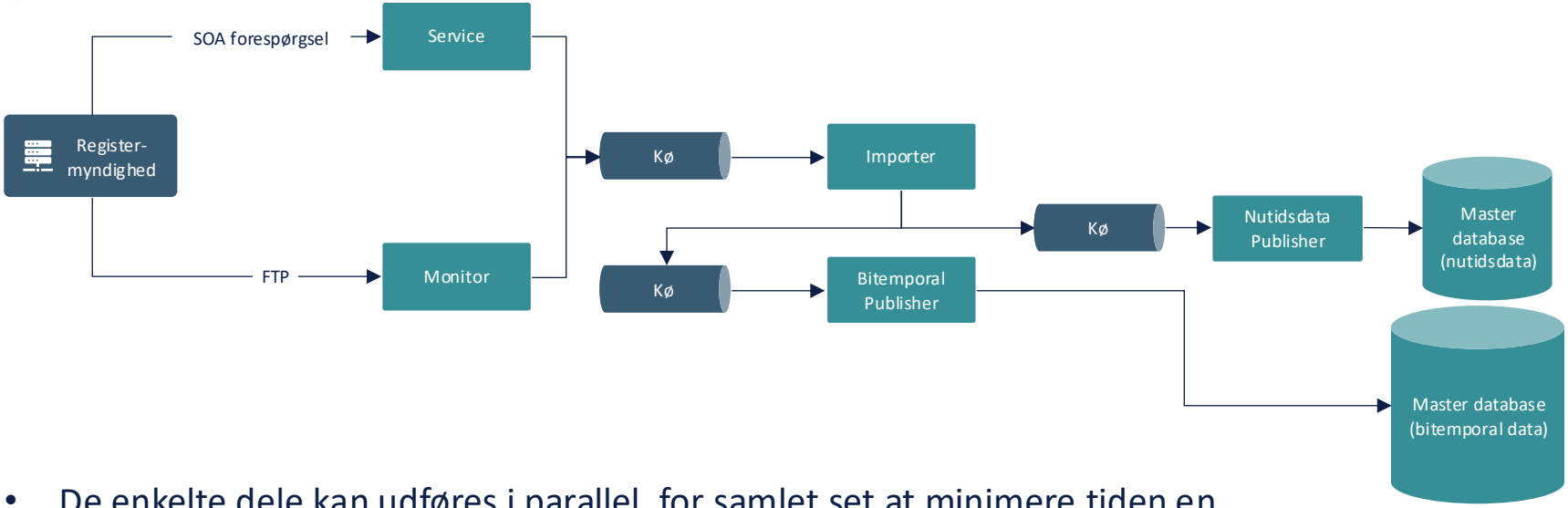
- Hvad er de største udfordringer ved den nuværende indlæsning for jer som anvendere/registre?

- Hvilke valideringer kan DAF tillade sig at lave på indlæsningstidspunktet og hvem skal give input til de regler?

- Er der ønske om mere indsigt i status på indlæsning, og i så fald, hvilken indsigt mangles?
  - F.eks. Statistik for antal indlæste rækker, kø-størrelser i løbet af dagen, gennemsnitlig tid fra pakke modtaget til fuldt indlæst?
  - Kan status på indlæsning være delvist offentligt tilgængeligt?

VISION FOR MODERNE INDLÆSNING

Vision for parallel indlæsning



- De enkelte dele kan udføres i parallel, for samlet set at minimere tiden en opdatering tager.

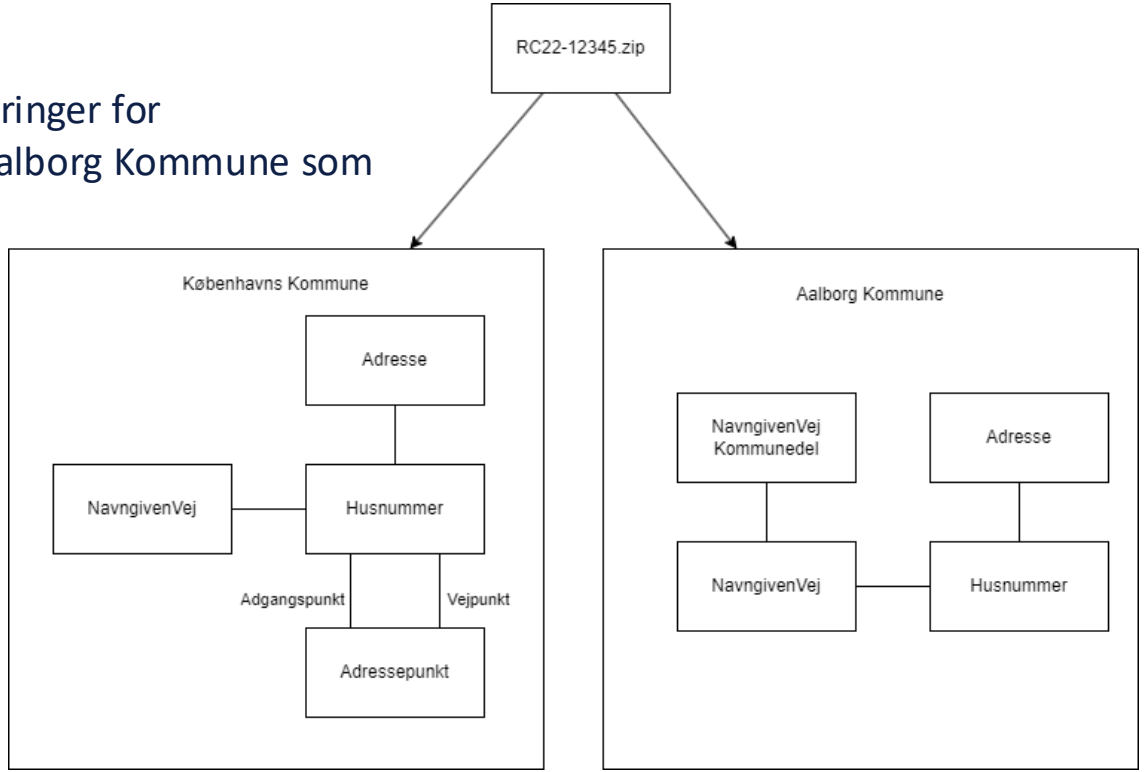
Vision om moderne indlæsningsflow

1. Register uploader zip-fil til SFTP server eller sender XML til SOAP service.
2. DAF ***paralleliserer*** udføring af skema-validering på den indsendte register-xml.
3. DAF ***paralleliserer*** indlæsning af register-xml til masterload database.
4. DAF ***paralleliserer*** overføring af indlæst data fra masterload til masterpublish database.
5. ~~DAF overfører indlæst data fra masterload til S0 masterpublish database~~
   - S0 zonen udfases, så dette step udgår.
6. ~~DAF genererer hændelser i masterload database~~
   - Generering af personlige hændelser udgår, fordi moderne hændelser ligger direkte på data
7. DAF meddeler registret at indlæsningen er færdig.
8. PULL hændelser er klar til at blive hentet ***via moderne tjenester.***
9. ~~DAF sender PUSH-hændelser til abonnenter.~~
   - PUSH-hændelser udfases.

Forventninger

- Man kan inddele pakkerne i mindre dele som hver især kan processeres parallelt.

- F.eks. pakke indeholder ændringer for Københavns Kommune og Aalborg Kommune som ingen afhængigheder har.



Parallelisering af indlæsning til masterload (1/4)

- Der vil for så vidt muligt ikke ske nogen ændringer til snitfladen for anvendere.

- Men registre kan tilvælge paralleliseret indlæsning på RC niveau.
  - Således kan f.eks. DAR og BBR indlæses parallelt, mens MAT fortsat håndteres sekventielt.

Parallelisering af indlæsning til masterload (2/4)

- Indlæsningen må parallelisere individuelle xml filer i samme zip fil.

RC22-2769826.zip

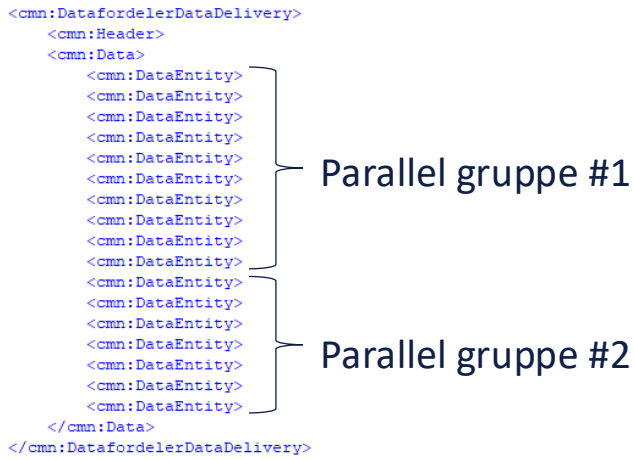
Navn	Størrelse	Pakket størrelse
Københavns Kommune.xml	26 629	1 289
Aalborg Kommune.xml	7 661	1 128

```
<cmn:DatafordelerDataDelivery>
    <cmn:Header>
    <cmn:Data>
        <cmn:DataEntity>
        <cmn:DataEntity>
        <cmn:DataEntity>
        <cmn:DataEntity>
        <cmn:DataEntity>
        <cmn:DataEntity>
        <cmn:DataEntity>
        <cmn:DataEntity>
        <cmn:DataEntity>
        <cmn:DataEntity>
        <cmn:DataEntity>
        <cmn:DataEntity>
        <cmn:DataEntity>
        <cmn:DataEntity>
        <cmn:DataEntity>
        <cmn:DataEntity>
        <cmn:DataEntity>
        <cmn:DataEntity>
    </cmn:Data>
</cmn:DatafordelerDataDelivery>
```

2x

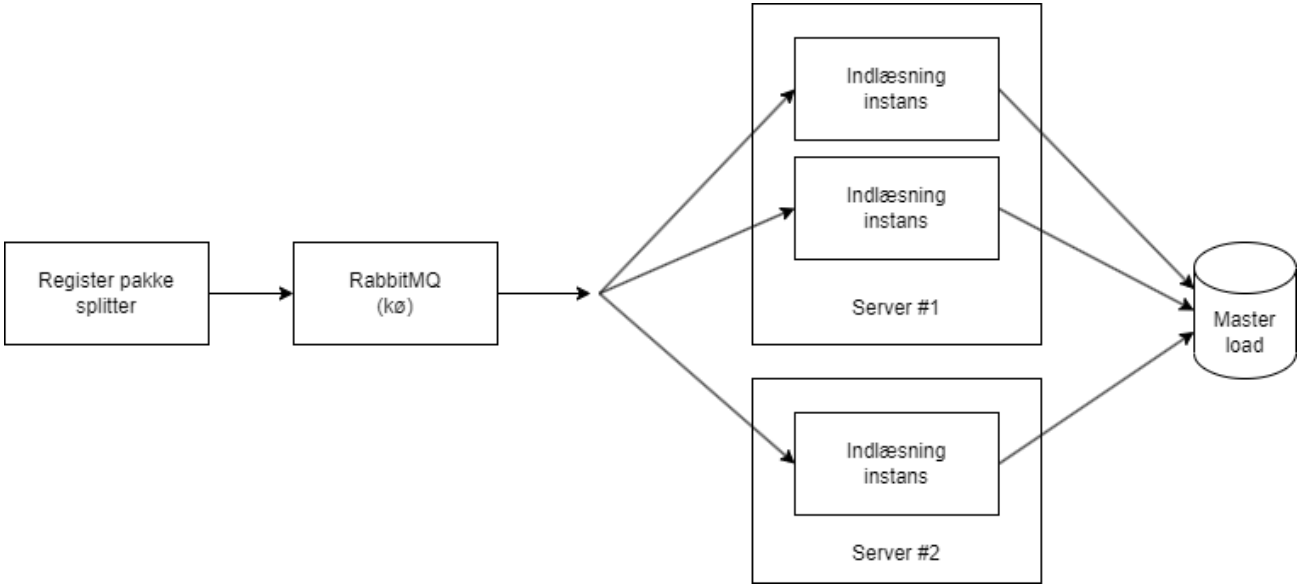
Parallelisering af indlæsning til masterload (3/4)

- Registeret kan fremsende en *paralleliseringsnøgle* for indholdet af deres fil, hvor registret garanterer at indhold med forskellige paralleliseringsnøgler ikke ændrer samme data.
- Hvordan kan den nøgle se ud?
  - Option #1, simpel: Registret siger at nøglen er ”id_lokalId”, så alle ændringer til samme objekt vil blive indlæst i samme gruppe.
  - Option #2, kompleks: Registret tilføjer et nyt felt til deres skema, f.eks. ”parallel_key”, og sender sammenhængende entiteter for forskellige tabeller med samme ”parallel_key”. Disse vil alle blive indlæst i samme gruppe.



Parallelisering af indlæsning til masterload (4/4)

- Alle parallelle grupper lægges på en kø.
- Beskeder fra køen processeres parallelt via *competing consumers* som kan skaleres.



TOTALINDLÆSNING I DAG

Hvordan virker totalindlæsning i dag (tabuler data)

1. Registeret og DAF/SDFI aftaler at der vil blive udført en totalindlæsning
2. Registeret fremsender et totaludtræk af data.
3. DAF sletter registerets data fra masterload.
4. DAF udfører indlæsning af registrets totaludtræk til masterload.
5. DAF overfører data fra masterload til skyggetabeller i masterpublish.
6. Når indlæsningen af skyggetabeller i masterpublish er færdig, udskiftes de rigtige tabeller med skyggetabelerne.
   - Hermed er registrets data kun utilgængeligt i få sekunder / minutter.
7. Totalindlæsningen er færdig.
8. Delta-indlæsning genoptages.

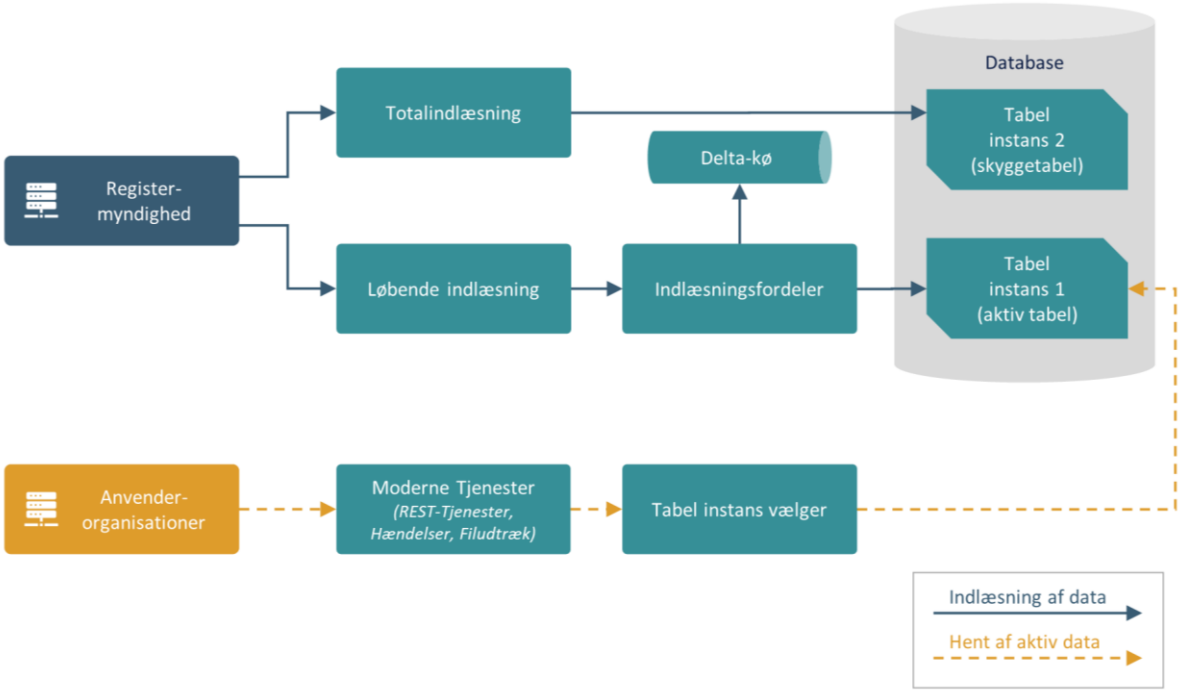
VISION FOR OPTIMERET TOTALINDLÆSNING

Dialog

- Såfremt at DLS processen vedrørende opdatering af tjenester og dertilhørende totalindlæsninger var mere smidig, ville I så gøre brug af dette oftere?

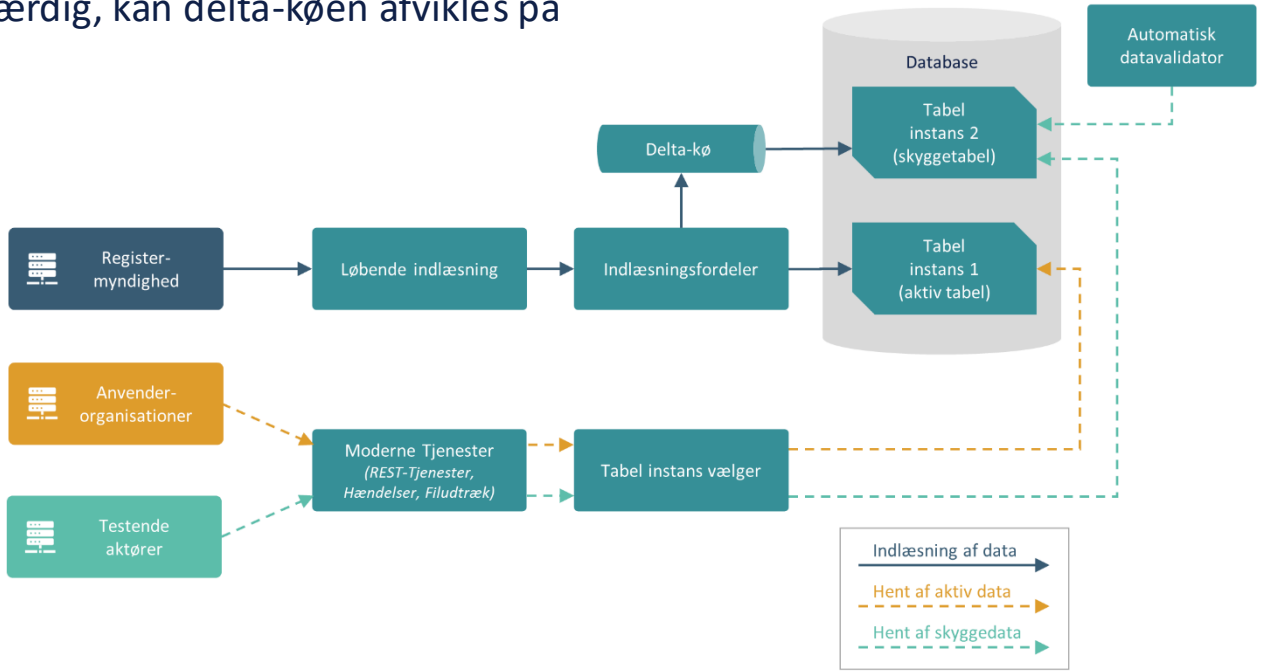
Forbedringer til totalindlæsning flowet

- Den løbende delta-indlæsning kører mens totalindlæsningen foregår.
  - Dette kræver at der indlæses til en skyggetabel både i masterload og masterpublish



Verificering og test af totalindlæsning

- Når totalindlæsningen er færdig, kan delta-køen afvikles på skyggetabellen.
- Efter totalindlæsning kan testende aktører validere data i skyggetabelen inden denne er tilgængelig for anvendere.



Totalindlæsning og moderne hændelser

- Moderne hændelser skal også understøtte totalindlæsninger.
- Der er 3 scenarier vedrørende hændelser:
  - Der skal laves hændelser for samtlige rækker i totaludtrækket.
  - Der skal laves hændelser kun for ændrede rækker i totaludtrækket.
  - Der skal ikke laves nogen hændelser.
    - Kunne evt. være relevant hvis registeret har tilføjet et nyt felt i deres datamodel.
- Visionen er at DAF understøtter alle 3 scenarier, og at registeret for hver totalindlæsning kan vælge hvilken de ønsker skal bruges denne gang.
  - Det vil være registerets opgave at meddele til anvendere hvordan registeret ønsker de skal håndtere denne, på baggrund af det valgte hændelse scenarie.

DATAMODEL VERSIONERING I DAG

Datamodel versionering i dag

- Der er ingen datamodel versionering i dag. Der er udelukkende tjeneste-versionering som hører til en specifik DLS version.

- Når registrene ønsker ændringer i deres datamodel, er der 2 muligheder:
  - Hvis ændringerne er 'små nok' kan de indarbejdes i de eksisterende tjenester på en bagud-kompatibel måde.
    - Hvis ændringerne er små men ikke bagud-kompatible ifølge tjenestens skema er der blevet opsat nye tjeneste-versioner.
  - Hvis ændringerne er store, bliver der opsat en ny replikeringskanal, ligesom hvis der kom et helt nyt register.
    - Her må registeret så sende data i nyt og gammelt format indtil den gamle version udfases.

VISION OM DATAMODEL VERSIONERING

Datamodel versionering

- Moderne DAF vil understøtte versionering på både tjeneste og datamodel niveau.
  - I dag snakker vi udelukkende om datamodel versionering.

- Mulighed for simpel versionering af datamodeler og paralleldrift af nye og gamle tjenester vil nødvendigvis være fuldt ud afhængig af omfanget af registrenes ændringer.

Datamodel versionering

- Der er 3 forskellige scenarier:
  1. Felter/tabeller er blevet tilføjet
  2. Felter/tabeller er blevet omdøbt
  3. Felter/tabeller er blevet fjernet

- Gamle versioner af tjenester udfases i dag typisk efter 6 måneder. Hvor længe kan der gå inden gamle datamodel versioner udfases?

Datamodel versionering – bagud kompatibilitet

- Med Moderne Tjenester kan anvendere selv angive deres joins på tværs af register tabeller.
  - Datamodellen skal dermed udstilles som en samlet versioneret model.
  - En ikke-bagud-kompatibel ændring til et specifikt register kan dermed få konsekvenser for mange anvendere, da versionering ikke kan ske på register-niveau.
    F.eks. DAR laver en ændring til Husnummer-tabellen, og vi har en anvender som har en sammensat forespørgsel som joiner fra CPR -> EJF -> BBR -> DAR

- Udgangspunkt i 'DAF Grunddatamodel' version.
  - Første version af de Moderne Tjenester vil have 'DAF Grunddatamodel' versionsnummer 1.
  - Hvis DAR har brug for at lave en ikke-bagud-kompatibel ændring skal Moderne Tjenester datamodellen opdateres, og får dermed versionsnummer 2.
    - Alle anvendere skal dermed opdatere det versionsnummer de gør brug af, også hvis de aldrig bruger DAR. Opdatering til denne nye version vil dog være triviel i deres situation.

Datamodel versionering – Nye felter/tabeller

- Tilføjelse af nye felter til snitfladen er en bagud kompatibel ændring, og kræver dermed ikke en ny version af den samlede datamodel.
  - De Moderne Tjenesters skema-definition vil tillade dette.
- Kan dermed udføres uden nævneværdige udfordringer.

Datamodel versionering – Felter/tabeller omdøbes

- Omdøbning af felter er ikke bagud kompatibelt. Der skal laves en ny datamodel version.
- Anvendere som henter den nye datamodel version får de nye navne.
- Anvendere som henter den gamle datamodel version får stadig de gamle navne.
  - Registeret skal, i forbindelse med datamodelopdateringen, blot levere en mapningsmodel som beskriver hvordan disse skal mappes.

Husnummer v1

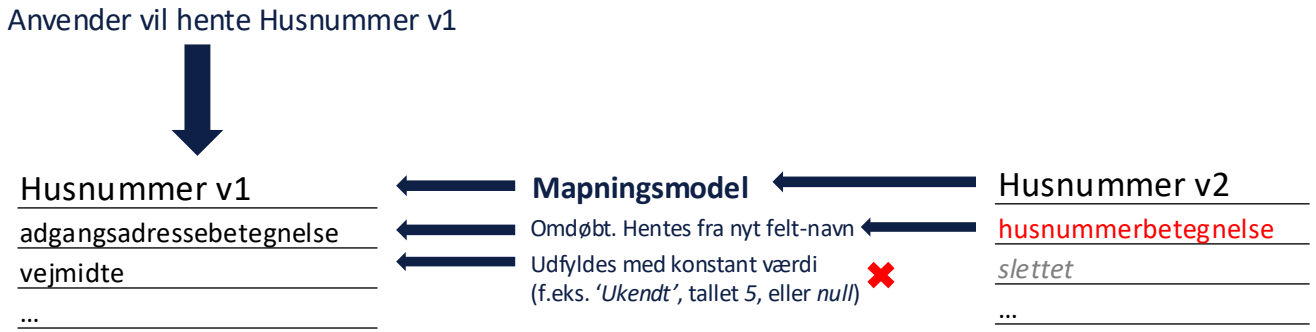
id_lokalId	adgangsadressebetegnelse	…
1ad8csd-…	Strandgade 3, 1401 København K	…
54635c-…	Grønningen 17, 1270 København	…
9832bd-…	Rentemestervej 8, 2400 København	…

Husnummer v2

id_lokalId	husnummerbetegnelse	…
1ad8csd-…	Strandgade 3, 1401 København K	…
54635c-…	Grønningen 17, 1270 København	…
9832bd-…	Rentemestervej 8, 2400 København	…

Datamodel versionering – Felter/tabeller fjernes

- Fjernelse af felter er ikke bagud kompatibelt.
- Hvis registeret fjerner gamle felter i en datamodelændring skal den leverede mapningsmodel specificere hvordan felterne udfyldes i udstillingen af tilbagekonverteret data.



Datamodel versionering – Felter/tabeller ændres

- Det er simpel mapning som vil være understøttet i tilbagekonverteret datamodeller.
  - Tunge beregninger som f.eks. indeholder joins vil ikke være tilladt.
  - Det vil ikke være muligt for anvendere at lave filtrering på beregnede felter, da dette ikke kan indekseres i databasen.
- Hvis registeret har brug for mere komplekse beregninger kan en mapningsmodel ikke understøtte dette, og en anden løsning skal bruges hvis bagud-kompatibilitet skal sikres.
  - En ny replikeringskanal kan etableres, ligesom sker i dag for store datamodel ændringer
  - Registeret *skal* sende data i begge versioner under parallel-driften.
  - Hvis ændringerne er begrænset til en enkel tabel, kan registeret fremsende samme tabel 2 gange, i det gamle og nye format.


netcompany
www.netcompany.com