Datafordeler

Dataleverancespecifikation

Implementeringsdetaljer

Version 1.0.0

2016 09 01



# Versions historik

|  |  |  |
| --- | --- | --- |
| **Dato** | **Version** | **Bemærkning** |
| 01-09-2016 | 1.0 | Første version af dokumentet |

Indholdsfortegnelse

[Versions historik 2](#_Toc461803300)

[Introduktion 4](#_Toc461803301)

[Om dette dokument 4](#_Toc461803302)

[Intervalopslag 5](#_Toc461803303)

[Input 5](#_Toc461803304)

[Output 5](#_Toc461803305)

[Bitemporal join eksempel 6](#_Toc461803306)

[MedDybde parameter 11](#_Toc461803307)

[Joins 13](#_Toc461803308)

[Bitemporale joins 13](#_Toc461803309)

[DAGI joins 13](#_Toc461803310)

# Introduktion

### Om dette dokument

Dette dokument udvider de logiske beskrivelser i de øvrige bilag, især bilag 2, med mere tekniske forklaringer af potentielt særligt komplekse aspekter. Herudover så kan dette dokument også indeholde eksempler på, hvordan en fysisk implementering af logikken fra de andre bilag kan realiseres i henhold til de logiske beskrivelser og de fysiske data.

# Intervalopslag

I bilag 2 og bilag 2b er der specificeret henholdsvis REST og filudtræk tjenester – alle med mulighed for opslag på intervaller i tidsdimensionerne – virkningstid, registreringstid og DAF opdateringstid.

For filudtræk tjenesterne er disse intervalopslag ligetil, da der ingen joins foretages og der blot returneres data fra en enkelt tabel ad gangen. Dette afsnit er således ikke relevant for filudtræk intervalopslag, som blot skal fungere præcis som specificeret i bilag 2b.

For REST tjenesterne foretages der dog joins med andre tabeller, hvis data returneres som nestede objekter. Der er i dette afsnit forklaret, hvordan dette tænkes realiseret.

## Input

For ikke at gøre intervalopslag unødvendigt komplekst, men stadig understøtte det kritiske behov der er i forbindelse med hændelsesprocessering, så vil det være okay, hvis der kun understøttes intervalopslag på en tidsdimension ad gangen og altid sammen med et ID for det ønskede objekt. Dette betyder for eksempel, at når der modtages en hændelse om at et objekt er ændret, giver et intervalopslag mulighed for at hente hvordan objektet ser ud i alle dets nye virkningsperioder ud fra dets ID og et registreringstidspunkt.

Input skal for intervalopslag således altid indeholde:

* Et ID på hovedentitet der spørges på:
  + Adresse
  + Husnummer
  + NavngivenVej
  + NavngivenVejKommunedel
  + Postnummer
  + SupplerendeBynavn
* Et interval for enten registreringstid (RegistreringFra og RegistreringTil), virkningstid (VirkningFra og VirkningTil) eller DAF opdateringstid (DAFTimestampFra og DAFTimestampTil).
  + Der må kun angives et interval for én af tidsdimensionerne i et intervalopslag.
* For tidsdimensioner, der ikke er angivet et interval for, kan der specificeres et tidspunkt ved at sætte fra- og til-tidspunkt til samme værdi.
  + Fx hvis der er efterspurgt et virkningsinterval, så kan RegistreringFra og RegistreringTil sættes til et tidspunkt, hvor der som standard ellers ville blive brugt ”NOW( )”.

## Output

Resultatet af et intervalopslag består af samtlige konfigurationer som den returnerede objektstruktur har haft i det forespurgte interval. Dette er simpelt for et objekt, der ingen nestede objekter har – da der således blot returneres de versioner af objektet der har eksisteret inden for det forespurgte interval.

Når et objekt har nestede objekter, så er disse med til at definere hvordan den fulde returnerede objektstruktur ser ud. Hvis et nestet objekt har ændret sig i det efterspurgte interval, så vil dette skulle medregnes når resultatet for intervalopslaget udvælges.

En måde at forstå dette på, er ved at kombinere samtlige perioder for hovedobjektet og alle dets underobjekter på en samlet tidslinje. Herefter kan der så laves enkeltforespørgsler for hver eneste af disse resulterende perioder. Dette er illustreret i nedenstående figur.



Denne metode, med først at finde perioder der herefter laves punktforespørgsler på, er der givet et T-SQL eksempel på i følgende afsnit.

## Bitemporal join eksempel

Dette afsnit introducerer et T-SQL eksempel på hvordan bitemporale joins kan implementeres. Der er mange måder at implementere dette på, så samme resultater findes, og dette er blot et eksempel på hvordan det kan gøres.

Udgangspunktet for eksemplet er Husnummer tjenesten, og der benyttes følgende inputparametre:

* RegistreringFra: '2017-01-01 00:00:00'
* RegistreringTil: '2017-01-01 00:00:00' (samme som RegistreringFra)
* VirkningFra: '2008-01-01 00:00:00'
* VirkningTil: '2016-01-01 00:00:00'
* Id: '2EDDA757-93E0-1605-E044-0003BA298018'

Husnummer servicen returnerer følgende objekter:

* Husnummer
  + Adgangspunkt
  + Vejpunkt
  + Postnummer
  + SupplerendeBynavn
  + Kommuneinddeling
  + Meninghedsrådsafstemningsområde
  + Afstemningsområde
  + Sogneinddeling
  + AdresseList
  + NavngivenVej
    - NavngivenVejKommunedelList

Den her eksemplificerede tilgang finder alle de timestamps hvor ændringer i objektstrukturen er trådt i kraft inden for det opslåede interval. Dette er her vist for et virkningstid intervalopslag, med ét registreringstidspunkt – men kan også benyttes for andre intervalopslag.

Følgende query benytter data fra et tidligere testdatasæt, så der kan være sket ændringer i data efterfølgende, således at querien måske ikke returnerer det samme, hvis kørt mod nyt testdata. Herudover så er querien kørt mod den interne DAR model, som ikke stemmer 1:1 overens med DARs DAF model.

DECLARE @VirkningFraInput datetime2

DECLARE @VirkningTilInput datetime2

DECLARE @RegistreringTid datetime2

set @VirkningFraInput = '2008-01-01 00:00:00'

set @VirkningTilInput = '2016-01-01 00:00:00'

set @RegistreringTid = '2017-01-01 00:00:00'

DECLARE @HusnummerId uniqueidentifier

set @HusnummerId = '2EDDA757-93E0-1605-E044-0003BA298018'

select \* from

(select Virkningstider.value('(text())[1]', 'datetime2') as VirkningsTid from

(select

Husnummer.VirkningFra

,Husnummer.VirkningTil

,Adresse.VirkningFra

,Adresse.VirkningTil

,Adgangspunkt.VirkningFra

,Adgangspunkt.VirkningTil

,Vejpunkt.VirkningFra

,Vejpunkt.VirkningTil

,NavngivenVej.VirkningFra

,NavngivenVej.VirkningTil

,NavngivenVejKommunedel.VirkningFra

,NavngivenVejKommunedel.VirkningTil

,Postnummer.VirkningFra

,Postnummer.VirkningTil

,SupplerendeBynavn.VirkningFra

,SupplerendeBynavn.VirkningTil

,DARAfstemningsområde.VirkningFra

,DARAfstemningsområde.VirkningTil

,DARMenighedsrådsafstemningsområde.VirkningFra

,DARMenighedsrådsafstemningsområde.VirkningTil

,DARSogneinddeling.VirkningFra

,DARSogneinddeling.VirkningTil

,DARKommuneinddeling.VirkningFra

,DARKommuneinddeling.VirkningTil

from Husnummer

left join Adresse

on Husnummer.Id = Adresse.Husnummer\_id

and Adresse.RegistreringFra <= @RegistreringTid and ( Adresse.RegistreringTil is null or Adresse.RegistreringTil > @RegistreringTid)

left join Adressepunkt Adgangspunkt

on Husnummer.Adgangspunkt\_id = Adgangspunkt.Id

and Adgangspunkt.RegistreringFra <= @RegistreringTid and ( Adgangspunkt.RegistreringTil is null or Adgangspunkt.RegistreringTil > @RegistreringTid)

left join Adressepunkt Vejpunkt

on Husnummer.Vejpunkt\_id = Vejpunkt.Id

and Vejpunkt.RegistreringFra <= @RegistreringTid and ( Vejpunkt.RegistreringTil is null or Vejpunkt.RegistreringTil > @RegistreringTid)

left join NavngivenVej

on Husnummer.NavngivenVej\_id = NavngivenVej.Id

and NavngivenVej.RegistreringFra <= @RegistreringTid and ( NavngivenVej.RegistreringTil is null or NavngivenVej.RegistreringTil > @RegistreringTid)

left join NavngivenVejKommunedel

on NavngivenVej.Id= NavngivenVejKommunedel.NavngivenVej\_id

and NavngivenVejKommunedel.RegistreringFra <= @RegistreringTid and ( NavngivenVejKommunedel.RegistreringTil is null or NavngivenVejKommunedel.RegistreringTil > @RegistreringTid)

left join Postnummer

on Husnummer.Postnummer\_id= Postnummer.id

and Postnummer.RegistreringFra <= @RegistreringTid and ( Postnummer.RegistreringTil is null or Postnummer.RegistreringTil > @RegistreringTid)

left join SupplerendeBynavn

on Husnummer.SupplerendeBynavn\_id= SupplerendeBynavn.id

and SupplerendeBynavn.RegistreringFra <= @RegistreringTid and ( SupplerendeBynavn.RegistreringTil is null or SupplerendeBynavn.RegistreringTil > @RegistreringTid)

left join DARKommuneinddeling

on Husnummer.DARKommune\_id= DARKommuneinddeling.id

and DARKommuneinddeling.RegistreringFra <= @RegistreringTid and ( DARKommuneinddeling.RegistreringTil is null or DARKommuneinddeling.RegistreringTil > @RegistreringTid)

left join DARMenighedsrådsafstemningsområde

on Husnummer.DARMenighedsrådsafstemningsområde\_id= DARMenighedsrådsafstemningsområde.id

and DARMenighedsrådsafstemningsområde.RegistreringFra <= @RegistreringTid and ( DARMenighedsrådsafstemningsområde.RegistreringTil is null or DARMenighedsrådsafstemningsområde.RegistreringTil > @RegistreringTid)

left join DARAfstemningsområde

on Husnummer.DARAfstemningsområde\_id = DARAfstemningsområde.id

and DARAfstemningsområde.RegistreringFra <= @RegistreringTid and ( DARAfstemningsområde.RegistreringTil is null or DARAfstemningsområde.RegistreringTil > @RegistreringTid)

left join DARSogneinddeling

on Husnummer.DARSogneinddeling\_id = DARSogneinddeling.id

and DARSogneinddeling.RegistreringFra <= @RegistreringTid and ( DARSogneinddeling.RegistreringTil is null or DARSogneinddeling.RegistreringTil > @RegistreringTid)

where Husnummer.Id = @HusnummerId and Husnummer.RegistreringFra <= @RegistreringTid and ( Husnummer.RegistreringTil is null or Husnummer.RegistreringTil > @RegistreringTid)

FOR XML PATH(''), TYPE

) as sub(VirkningsTid)

CROSS APPLY

VirkningsTid.nodes('/\*') AS VirkningsTider(Virkningstider)

) as Result

where VirkningsTid >= @VirkningFraInput and (VirkningsTid < @VirkningTilInput or @VirkningTilInput is null)

group by VirkningsTid

Denne query resultere i følgende liste af virkningstider for den samlede objektstruktur:

|  |
| --- |
| Virkningstider |
| 2009-11-24 01:50:11.000 |
| 2010-12-20 16:05:23.807 |
| 2012-03-07 14:53:59.787 |
| 2012-03-07 14:54:00.077 |
| 2012-03-08 09:56:23.207 |

På baggrund af ovenstående virkningstider for den samlede struktur, samt inputparametre for virkningstid, kan der nu slås op hvordan hovedobjektet, med dets underobjekter så ud inden for hver af de fundne perioder. Der er tale om følgende perioder, for hvilke der skal laves et punktopslag indenfor (hvor fra-tidspunktet fordelagtigt kan benyttes).

|  |  |
| --- | --- |
| Fra (inklusiv) | Til (eksklusiv) |
| 2008-01-01 00:00:00.000 | 2009-11-24 01:50:11.000 |
| 2009-11-24 01:50:11.000 | 2010-12-20 16:05:23.807 |
| 2010-12-20 16:05:23.807 | 2012-03-07 14:53:59.787 |
| 2012-03-07 14:53:59.787 | 2012-03-07 14:54:00.077 |
| 2012-03-07 14:54:00.077 | 2012-03-08 09:56:23.207 |
| 2012-03-08 09:56:23.207 | 2016-01-01 00:00:00.000 |

For disse enkeltopslag skal VirkningTil og VirkningFra begge være det fundne tidspunkt (fra-tidspunkt), mens at RegistreringTil og RegistreringFra begge skal sætte til samme værdi som tidligere nævnt. I det tilfælde hvor hovedobjektet først eksisterer senere end fra-tidspunktet for det efterspurgte interval, så vil de/det første enkeltopslag ikke returnere noget, hvilket er som forventet.

Med ovenstående metode findes således alle versioner for det efterspurgte objekt i det efterspurgte tidsinterval. Samme metode kan ligeledes benyttes til joins med registreringstidsintervaller.

For DAF opdateringsintervaller benyttes der ikke samme metode til join af data, da joins kun sker ud fra de bitemporale egenskaber.

# MedDybde parameter

Visse tjenester, der inkluderer nestede objekter, kan kaldes med en ”MedDybde” parameter, som bestemmer om hovedobjektets nestede objekter indeholder fuld data eller kun ID’er.

Hvis MedDybde er TRUE (eller NULL) så udfyldes JSON objektet med alt data, også nestede objekter; det vil sige data fra hovedobjektet samt relaterede objekters data. Hvis MedDybde er FALSE så indeholder JSON kun data som findes i hovedobjektet samt ID’er for hovedobjektets relaterede data.

Eksempel på et hovedobjekt med underobjekt der kaldes med dybde (underobjektets data markeret med rødt):

{

"id":"http://data.gov.dk/reg/hoved#00000000-0000-0000-0000-000000000001",

"id\_lokalId":"00000000-0000-0000-0000-000000000001",

"DATA":"HOVEDOBJEKT\_DATA",

"underobjekt":{

"id":"http://data.gov.dk/reg/under#00000000-0000-0000-0000-000000000002",

"DATA":"UNDEROBJEKT\_DATA"

}

}

Samme hovedobjekt, kaldt uden dybde:

{

"id":"http://data.gov.dk/reg/hoved#00000000-0000-0000-0000-000000000001",

"id\_lokalId":"00000000-0000-0000-0000-000000000001",

"DATA":"HOVEDOBJEKT\_DATA",

"underobjekt":{

"id":"http://data.gov.dk/reg/under#00000000-0000-0000-0000-000000000002"

}

}

Data i underobjektet, bortset fra ID’et, returneret således ikke. Dette gælder også underobjektets egne underobjekter, som heller ikke skal returneres.

Her er et tilsvarende eksempel, hvor der blot er en liste af underobjekter, i stedet for et enkelt.

Med dybde (underobjekterne markeret med rødt):

{

"id":"http://data.gov.dk/reg/hoved#00000000-0000-0000-0000-000000000001",

"id\_lokalId":"00000000-0000-0000-0000-000000000001",

"DATA":"HOVEDOBJEKT\_DATA",

"underobjektList":[

{

"id":"http://data.gov.dk/reg/under#00000000-0000-0000-0000-000000000002",

"underobjekt": {

"id":"http://data.gov.dk/reg/under#00000000-0000-0000-0000-000000000002"

"DATA":"UNDEROBJEKT\_DATA"

}

},

{

"id":"http://data.gov.dk/reg/under#00000000-0000-0000-0000-000000000003",

"underobjekt": {

"id":"http://data.gov.dk/reg/under#00000000-0000-0000-0000-000000000003"

"DATA":"UNDEROBJEKT\_DATA"

}

}

]

}

Uden dybde:

{

"id":"http://data.gov.dk/reg/hoved#00000000-0000-0000-0000-000000000001",

"id\_lokalId":"00000000-0000-0000-0000-000000000001",

"DATA":"HOVEDOBJEKT\_DATA",

"underobjektList":[

{

"id":"http://data.gov.dk/reg/under#00000000-0000-0000-0000-000000000002"

},

{

"id":"http://data.gov.dk/reg/under#00000000-0000-0000-0000-000000000003"

}

]

}

Således returneres kun underobjekternes ID’er uden nogen yderligere data.

# Joins

Tjenestelogik i bilag 2/2b er forsimplet med hensyn til filtre for joins, da de samme regler bør gælde for alle tjenester for alle registre. Der er i dette afsnit beskrevet, hvilke aspekter der skal tages højde for i forbindelse med joins.

## Bitemporale joins

Eftersom Grunddataprogrammet er bitemporalt, så vil al data også være bitemporalt. Dette betyder, at der ikke eksisterer konventionelle fremmednøgler, men at relationer mellem objekter derimod er underlagt bitemporalitet, og at relationer, ud over ID, også kræver et virkningstidspunkt og registreringstidspunkt når de slås op.

Samtlige joins angivet i tjenestelogikken i bilag 2/2b benytter samme bitemporale filtre, som selve hovedobjektet gør. De bitemporale filtre der er inkluderet for hovedobjektet i specifikationerne, skal ligeledes også benyttes til samtlige joins, inkluderet i samme tjenestelogik.

Fx så vil et hovedobjekt kunne slås op ud fra ID og med ”NOW( )” som både virkningstidspunkt og registreringstidspunkt, og alle joins som der laves i opslaget af dette hovedobjekt, vil således alle ske på baggrund af hvad der joines på i tjenestelogikken (forment ID/id\_lokalId) og yderligere med ”NOW( )” iht. virkningstid og registreringstid.

## DAGI joins

DAR har tværgående tjenester der joiner med DAGI data. Når der joines med DAGI objekter, så gøres dette i tjenestelogikken ved at joine på ”DAGI.Objekt.id\_lokalId” (tidligere beskrevet som ”DAGI.Objekt.id” – dog med samme mening).

DAR har kun begrænset indsigt i DAGI projektet og dettes model og data. Af denne grund, så kan DAR ikke med sikkerhed sige, hvad der skal til for at der på Datafordeleren kan joines med DAGI data. For at få dette bekræftet, er der brug for, at nogen med viden om både DAR, DAGI og DAF, kigger på data fra både DAR og DAGI i sammenhæng med specifikationerne for de tværgående DAR tjenester. Dette kan måske gøres i forbindelse med implementering på DAF, hvor det dog er vigtigt, at det gennemtænkes hvordan disse joins laves ud fra hvordan modeller og data ser ud.

Et par noter, det allerede er kendt, at der skal tages højde for i forbindelse med DARs joins med DAGI:

* Alle DARs referencer til ”DAGI” skal forstås som ”DAGIMulti”
* Joins med DAGI skal ske med samme bitemporale filtre, som også gælder for andre joins
* Joins med DAGI skal yderligere filtrere på DAGI skala = ”1:10.000”