Dimensionel modellering for Nedrivning til rapportbrug

Indhold

[Formål 3](#_Toc156886116)

[1. Teknisk design af databehandlingen 3](#_Toc156886117)

[1.1 Udtrækning 3](#_Toc156886118)

[1.2 Dimensionel modellering 4](#_Toc156886119)

[1.3 Historisk logning 4](#_Toc156886120)

[1.3.1 DW\_ValidFrom og DW\_ValidTo 4](#_Toc156886121)

[1.3.2 DW\_IsCurrent 5](#_Toc156886122)

[1.3.3 DW\_MostRecentOnday 5](#_Toc156886123)

# Formål

En velfungerende dimensionel datamodel for nedrivningsdata giver BYGST mulighed for at generere avancerede rapporter. Dette betyder, at data kan kombineres på en fleksibel måde, der understøtter organisationens forretningslogik, behov for beslutningsgrundlag, aktuelle indsigter og afrapportering.

# 1. Teknisk design af databehandlingen

Overordnet set består det tekniske design af databehandlingen af tre dele:

1. Først trækkes der rådata fra AppSmith ned på Nedrivning-databasen, hvorefter det overføres til Rapport-databasen på DW-CB.
2. Derefter bliver data bearbejdet for at danne en dimensionel datamodel, der imødekommer rapporteringsbehovet.
3. Resultatet er en datamodel, hvor tabellerne og kolonnerne har meningsfyldte navne og entiteterne er entydigt relateret til hinanden.

## 1.1 Udtrækning

Følgende tabeller hentes fra Nedrivning databasen, hvor der ikke er teknisk opsatte relationer mellem tabellerne og hvor data foreligger i normaliserede skemaer.

|  |
| --- |
| **Tabel navn** |
| Adresser |
| Brevproces |
| Faser |
| Intern\_screening |
| Maengdeopgoerelse |
| Miljoeproces |
| Miljoeraadgiverdelaftale |
| Minkavlere |
| Minkavlere\_Minkfarme |
| MinkavlersRaadgiver |
| Minkfarme |
| Minkfarme\_Faser |
| Minkfarme\_Processer |
| Myndighed |
| Nedriver\_Kontaktpersoner |
| Nedrivere |
| Nedrivergrupper |
| Nedrivergrupper\_Nedrivere |
| Nedrivningsproces |
| Nedrivningstilladelse |
| Projektledere |
| Sagsstatus |
| TekniskRaadgiver\_Kontaktpersoner |
| TekniskRaadgivere |
| TekniskRaadgivergrupper |

Dokumentationen dækker udelukkende hvordan data overføres til rapportlaget, og ikke hvordan rådata skabes i Nedrivning-databasen.

## 1.2 Dimensionel modellering

For alle tabellerne gælder det, at det skal være hurtigere og nemmere for brugeren at arbejde med data. Data fra enkelte tabeller samles og sammensættes på anden måde end i rådata, så forretningsprocesserne korrekt afspejles i de dimensionelle tabeller.

For hver eneste rapporteringsbehov/måling bygges der også en fact-tabel, som sammen med dimensions-tabellerne udgør et stjerne-skema. Dette skema er velegnet til brug for analyse- og rapporteringsbehov.

De normaliserede tabeller som er specificeret i afsnit 1.1 grupperes til større tabeller. Dette gøres for at gøre datamodellen mindre relevant for brugeren, samtidig med at antallet af joins minimeres.

*Dokumentationen for dimensionelle, fact og stjerneskemaet findes i GitHub / Draw.io.*

Når data overføres fra Nedrivning-databasen bibeholdes data så vidt som muligt medmindre det skal opfylde rapporteringsbehovet. Det gælder herunder navneændringer på kolonner og værdier, samtidig med at "DW" kolonnerne tilføjes (se afsnit 1.3). Der tilføjes derfor ikke nyt rådata.

## 1.3 Historisk logning

Der er blevet anvendt Type 2 SCD (Slowly Changing Dimensions) for at bevare hele historikken fordelt på respektive primær- og historiktabeller. Princippet for historiklogning er ensartet på tværs af tabellerne.

### 1.3.1 DW\_ValidFrom og DW\_ValidTo

Den historiske logning finder sted ved at der i hver række i tabellerne eksisterer et ”DW\_ValidFrom” og ”DW\_ValidTo” felt.

Hvis man eksempelvis har tre minkfarme vil data første gang ovenstående job kører komme til at se således ud:

|  |  |  |  |
| --- | --- | --- | --- |
| **CHR** | **Nedrivergruppe\_id** | **DW\_ValidFrom** | **DW\_ValidTo** |
| 93 | 1 | 01-01-2021 | 31-12-9999 |
| 253 | 2 | 01-01-2021 | 31-12-9999 |
| 88784 | 1 | 01-01-2021 | 31-12-9999 |

Lad os sige at en bruger den 10. januar ændrer. Næste gang vil indholdet af tabellen dermed blive:

|  |  |  |  |
| --- | --- | --- | --- |
| **CHR** | **Nedrivergruppe\_id** | **DW\_ValidFrom** | **DW\_ValidTo** |
| 93 | 1 | 01-01-2021 | 31-12-9999 |
| 253 | 2 | 01-01-2021 | 31-12-9999 |
| 88784 | 1 | **01-01-2021** | **10-01-2021** |
| 88784 | 2 | **10-01-2021** | **31-12-9999** |

Ved at denormalisere tabellerne samles historikken fra flere tabeller i én. I stedet for at inkludere alle dato kolonnerne fra tabellerne, samles værdierne af disse i en samlet kolonne.

Logikken for historikken formateres således, at

DW\_ValidFrom beholder pr. række den seneste ændring for de tilknyttede tabeller.

DW\_ValidTo beholder pr. række den tidligste ændring for de tilknyttede tabeller.

Ved at samle to tabeller og deres historik vil tabellen se således ud:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tabel 1 Gældende\_fra** | **Tabel 1**  **Gældende\_til** | **Tabel 2**  **Gældende\_fra** | **Tabel 2**  **Gældende\_til** | **DW\_ValidFrom DW\_ValidTo** |
| **01-01-2021** | **01-02-2021** | 01-01-2021 | 31-12-9999 | **01-01-2021** **01-02-2021** |
| **01-02-2021** | **01-03-2021** | 01-01-2021 | 31-12-9999 | **01-02-2021 01-03-2021** |
| **01-03-2021** | **31-12-9999** | 01-01-2021 | 31-12-9999 | **01-03-2021 31-12-9999** |

Konsekvensen af denne logik vil betyde, at man ikke får den seneste ændring for (i dette tilfælde) tabel 2, men at historikken fra tabel 1 kommer til at repræsentere rækken.

Dog ses der bort for denne logik, når der findes en tabel i rådata der afspejler processerne. Det skyldes, at denne tabel afspejler forretningslogikken. Der gemmes derfor ikke historik på opslagstabellerne, f.eks. når samme proces får skiftet navn (fra "Afslut" til "Afsluttet"), så anvendes seneste navneændring, som i det her tilfælde vil være "Afsluttet".

### 1.3.2 DW\_IsCurrent

For at identificere seneste registrering i data er der implementeret en logik i kolonnen "DW\_IsCurrent".

Hvis der sættes et filter på DW\_IsCurrent = '1' findes seneste registrering.

DW\_IsCurrent = '0' finder historiske registreringer.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tabel 1 Gældende\_fra** | **Tabel 1**  **Gældende\_til** | **Tabel 2**  **Gældende\_fra** | **Tabel 2**  **Gældende\_til** | **DW\_ValidFrom DW\_ValidTo DW\_IsCurrent** |
| 01-01-2021 | 01-02-2021 | 01-01-2021 | 31-12-9999 | 01-01-2021 01-02-2021 **0** |
| 01-02-2021 | 01-03-2021 | 01-01-2021 | 31-12-9999 | 01-02-2021 01-03-2021 **0** |
| 01-03-2021 | 31-12-9999 | 01-01-2021 | 31-12-9999 | 01-03-2021 31-12-9999  **1** |

For at kunne finde det aktuelle billede af data skal DW\_IsCurrent = '1' derfor altid være sat til '1' i alle tilknyttede dimensionstabeller.

### 1.3.3 DW\_MostRecentOnday

Ændringer bliver registreret helt ned på klokkeslættet, hvilket betyder, at der kan være flere ændringer i løbet af dagen. For, at opnå enighed om resultatet på dagen, er der implementeret en kolonne "DW\_MostRecentOnDay" der definerer den seneste ændring i løbet af dagen.

Eksempelvis når samme minkfarm har 4 forskellige ændringer i løbet af samme dag, så viser DW\_MostRecentOnDay = 1 (Den seneste ændring på dagen) DW\_MostRecentOnDay = 0 (De tidligere ændringer på dagen)

|  |  |  |  |
| --- | --- | --- | --- |
| **CHR** | **Nedrivergruppe\_id** | **DW\_ValidFrom** | **DW\_ValidTo DW\_MostRecentOnDay** |
| 88784 | 1 | 01-01-2021 01:00:00 | 10-01-2021 01:00:01 **0** |
| 88784 | 2 | 01-01-2021 01:00:01 | 10-01-2021 01:00:02 **0** |
| 88784 | 3 | 01-01-2021 01:00:02 | 10-01-2021 01:00:03  **0** |
| 88784 | 4 | 01-01-2021 01:00:03 | 10-01-2021 01:00:04 **1** |