LANDSBYGGEFONDEN – DCAB STATISTIK

# O0500 - SOFTWAREARKITEKTUR STATISTIKMODUL

| | |
|---|---|
| **Version:** | 1.0 |
| **Status:** | Final |
| **Approver:** | |
| **Author:** | [Author] |

netcompany

## Document history

| Version | Date | Author | Status | Comments |
|---|---|---|---|---|
| 0.1 | 30-05-2022 | | Draft | Filled out document |
| 0.2 | 30-05-2022 | | Draft | Added ETL design and Deployment |
| 1.0 | 01-07-2022 | | Final | Review |

## References

| Reference | Title | Author | Version |
|---|---|---|---|
| | | | |

# Table of contents



# 1 Introduction

This document describes the architecture of the DCAB Statistik module, and not the remainder of the solution including the data module and DCAB frontend.

# 2 Architectural goals and constraints

## 2.1 Architecture principles

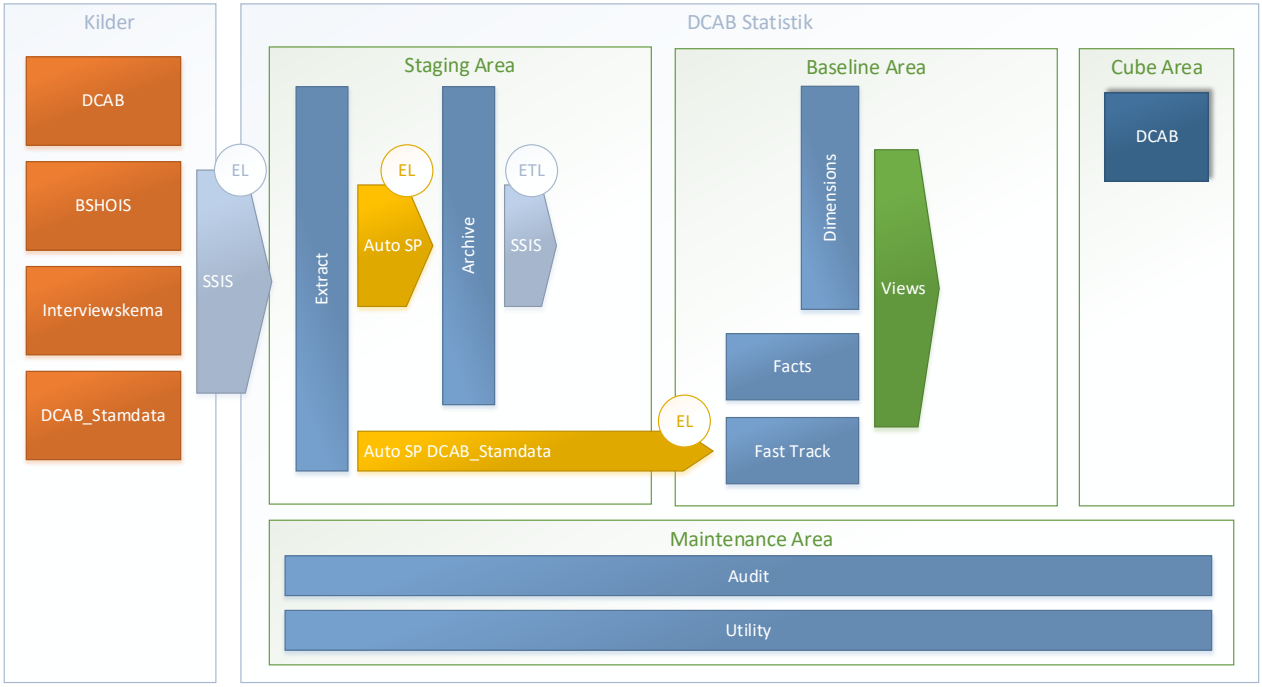
The data warehouse uses the operational data generated by the DCAB, Interviewskema and the BSHOIS database. None of the operational new data is generated or inserted directly into the data warehouse. The operational data is all sourced from the relevant database connections to the three systems.

# 3 Use case perspective

The advanced users are expected to have some technical and mathematics skills in order to filter and select the data and hence report on the data. The simple users however are expected to have a low degree of IT expertise, and the simple data view are made to support these users, so that they can get standard statistics without the need of IT expertise.

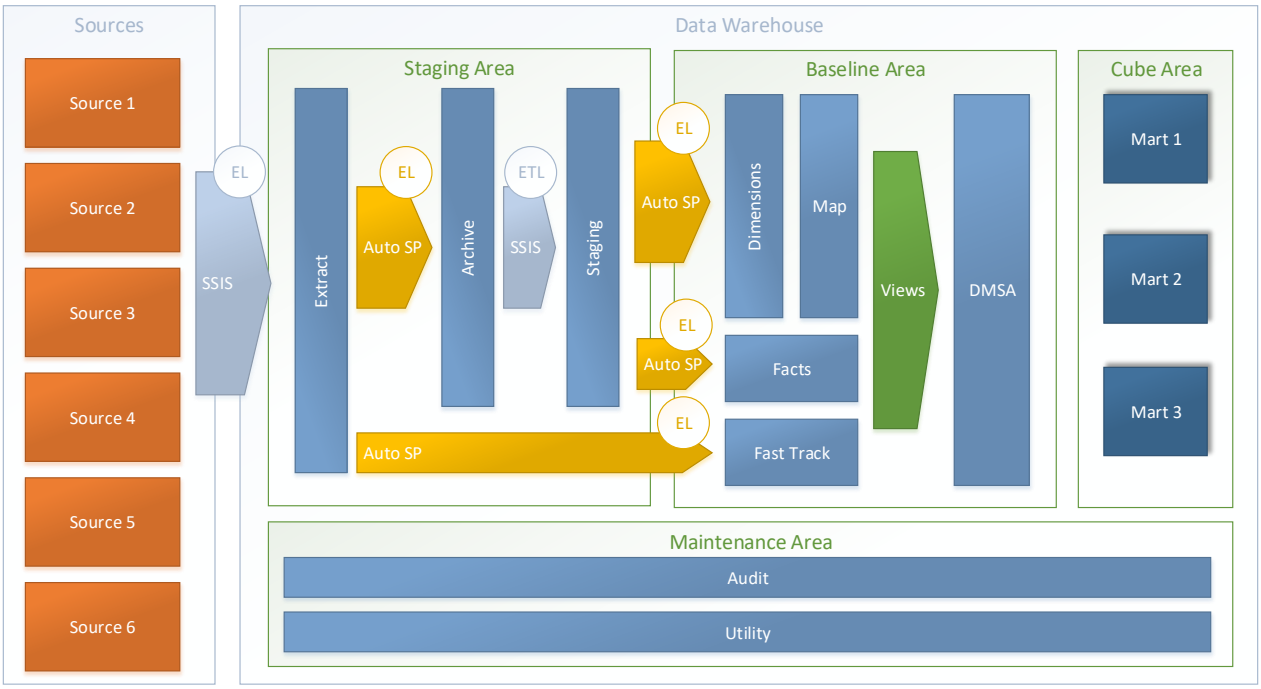
# 4 Implementation perspective

Please note that the DCAB Statistics project uses a variation of the project seen in the next documentation section, but the entire documentation is kept in case the need for these components arise at a later time. The following visualization shows what components are used in the DCAB Statistics project, the missing components are not implemented and are highlighted with a red heading if they are not used in the DCAB Statistics project.



# 4.1 Layers and components

The Data Warehouse architecture consists of the following logical layers as shown in the figure below i.e. the Staging Area, the Baseline Area, the Data Mart Area, the Maintenance Area and the Reporting layer.




Each layer is described in the subsequent sections in terms of their purpose as well as the policy applicable for each layer.

## 4.1.1 Staging Area

The purpose of the Staging Area is to receive data and store data from the sources including archiving of historical data. The Staging Area consists of 3 physical layers i.e. the Extract, the Archive and the Staging.

Data from the source systems are extracted and inserted into the Extract layer. The setup handles both Full loads and Delta loads from the sources. Physical deletion of data is marked with a flag in the Archive. Sources with Delta extract will require an extra ETL Flow to handle physical deletion, which loads all primary keys, then matches them against the Archive and marks all records which no longer exist in the source as deleted.

### 4.1.1.1 Extract

The Extract is used to receive data from the source systems with only meta data appended to it. This places a minimum of stress on the source system and makes the extracting process as fast as possible. When the data is moved to the extract tables, transformations can be done to it, without the data warehouse being limited by, or negatively impacting the source system.

### 4.1.1.2 Archive

The Archive is used to build history on the source data. Each record is identified by the business key as it is defined in the source. Any changes to the non-business key columns will then be stored with a row for each version of the record, and timestamps written in meta column, to indicate the time at which the version of the record was observed. The archive builds its history on based on the data in the extract tables, and no forms of transformations are done to the data.

Meta columns are also appended to each row. These indicate which rows are the current ones if the record has been deleted in the source and when that happened as well as information on the SSIS job that inserted the row.

### 4.1.2 Baseline Area

The Baseline area consists of several layers which are individually described in the following sections. All the business rules are implemented for the facts with SSIS. The data for the dimensions are in the staging tables where the business rules have already been applied to it. Stored procedures are used to update the dimensions with the data from their corresponding staging stables.

#### 4.1.2.1 Dimensions and Facts

The Dimensions and Facts layer contains the business relevant data in a structured and simplified manner. This makes it easy to use in a cube as well as for making ad hoc calculations. The data is implemented using a star schema / snowflake design following the Kimball approach to data modeling. Using this method, the facts contain data that can be aggregated, whereas the dimensions contain the data these aggregations can be filtered by.

#### 4.1.2.2 Fast Track

This area keeps data that does not require history, data which is already in the Baseline format or data that doesn't need the same attention to compliance and traceability as the other data in the baseline layer. It is also in this area which stored data that should be updated in near real time.

#### 4.1.2.3 DMSA

The Data Mart Staging Area (DMSA) is the area for staging shared dimension and fact tables used in different cubes or applications within or outside the Data Warehouse. Thus, this layer is also used to distribute data to other systems who subscribe to the information in any of the layers in the Data Warehouse. This area is made up of views rather than tables, to make the underlying data warehouse more flexible to change.

### 4.1.3 Maintenance Area

The Maintenance Area holds the two layers i.e. Audit and Utility. The Audit layer is used to hold data regarding execution and error logging. The execution log holds information about status on ETL runs, number of rows processed and data quality screening process. The Utility layer holds miscellaneous data and functionality, like fixed values and calendar generating functionality.

### 4.1.4 Cube Area

A tabular SSAS cube with a compatibility level of 1400 is chosen to run on a SQL Server 2017 solution. The cube should allow for different aggregates but any datatype or name changes are made directly in the Baseline area and not in the Cube.

## 4.2 ETL design

The solution is using an ETL design, which represents of three layers: Extract, Transform and Load. This design loads data from a source system into a datewarehouse, where the data can be mapped into fitting business rules and thereafter be loaded into a target, in the case a SSAS cube.

SSIS-packages are being used to extract data from the data sources and into extract tables. From there, stored procedures are executed to populate the archive tables. A second group of SSIS-packages are executed that takes the data from the archive tables and transforms it to the staging tables, which are loaded through views to the SSAS Cube.

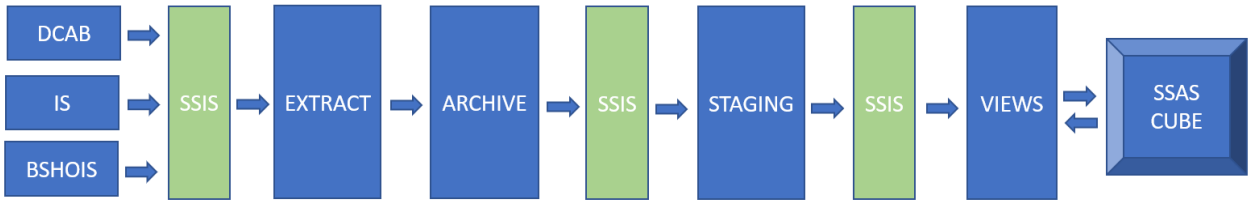



Figure 1 ETL Data flow

When the execute packages are started, they will audit the package execution start in tables in the audit schema.

The package allows for parallel execution with up to 4 threads, to define the number of threads there is a number parameter in the package that can be changed from 1-4. The default value is always 4 and the number can be changed upon execution of the SSIS package directly on the server its run.

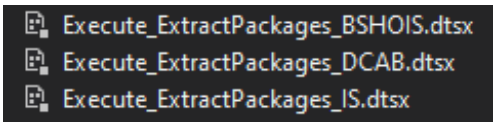



Figure 2 Execute packages

There are three data sources that we want to extract data from: DCAB, IS and BSHOIS. There has been created a SSIS extract package and an extract table to every considerable table in the three data sources. The SSIS packages all have equal set up. Here follow an example of this:

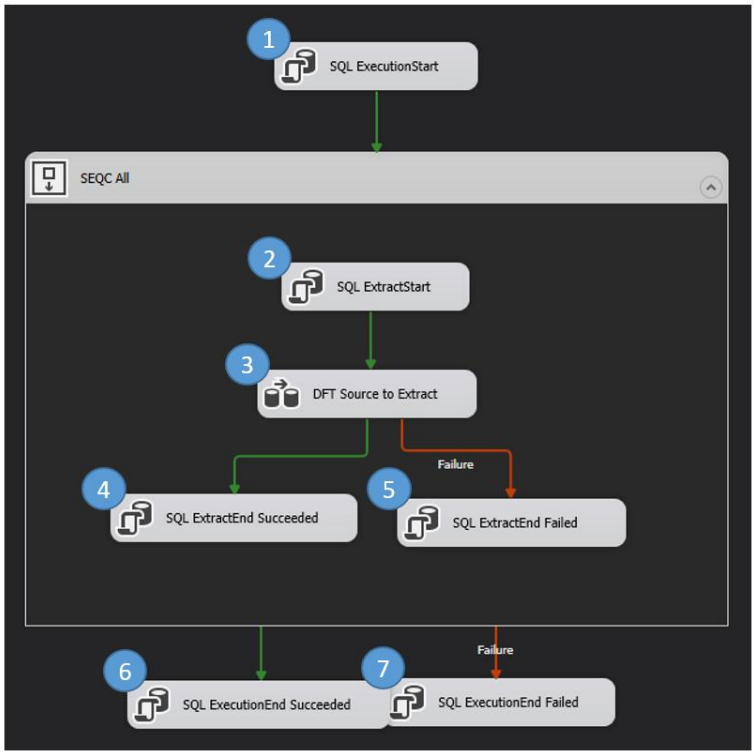



Figure 3 Control Flow

Description of steps:

1. **SQL Execution Start**: Start the logging procedure in the extract log.
2. **SQL ExtractStart:** Start the sequence.
3. **DFT Source to Extract:** Start dataflow (Figure 4).
4. **SQL ExtractEnd Succeeded:** The sequence succeed.
5. **SQL ExtractEnd Failed:** The sequence failed.
6. **SQL ExecutionEnd Succeded:** End the sequences and the logging procedure when succeded.
7. **SQL ExecutionEnd Failed:** End the sequences and the logging procedure when failed.

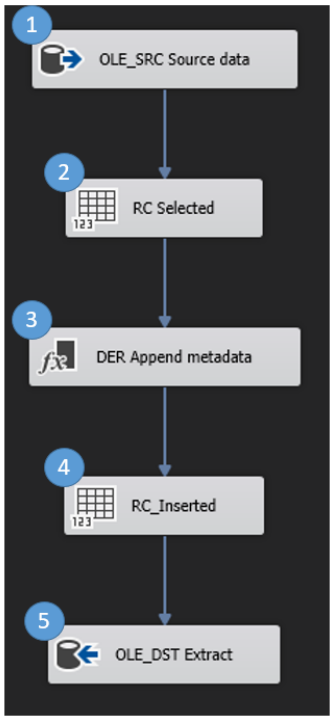



Figure 4 Data flow

Description of steps:

1. **OLE_SRC Source data:** Connects to the source table through OLE DB connection manager and uses a SQL query to get data result.
2. **RC Selected:** Metadata component that counts the amount of chosen rows by the SQL query.
3. **DER Append metadata:** Transformation component appends two metadata columns to the extract table, Meta_CreateJob and Meta_CreateTime.
4. **RC_Inserted:** Metadata component that counts the amount of rows that should be inserted into the extract table.
5. **OLE_DST Extract:** Connects to the destination table using OLE DB connection manager, map the columns and load the data result into the chosen extract table.

The execute archieve packages are using stored procedures that retrieve the data from the extract table and populates the archieve table.

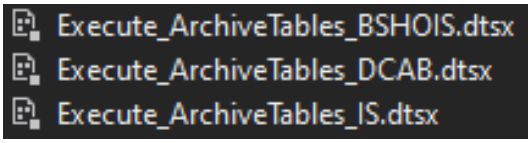



Figure 5 Execute archive packages

The UpdateArchieve is a procedure that checks if the data from an extract table exists in the corresponding archive table. From the archive tables, the data is transformed to fit business roles, and inserted into the staging tables. The staging table prepare the data before it loads to the baseline area. From the baseline area, the data is loaded into the SSAS cube.

To monitor the process, there has also been implemented audit functionality. These are stored procedures (figure 5) that logs every time data is moved from one table to another. The audit log tables records metadata such as package name and start time, and also data about how many rows that were selected.

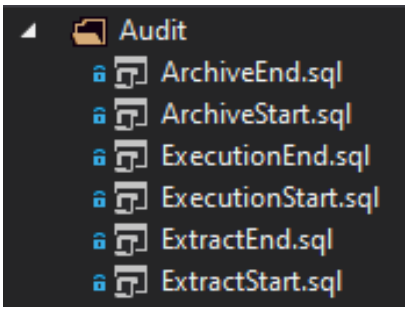



Figure 6 Audit procedures

# 5 Deployment perspective

Everytime the master/main branch is updated, a deployment is executed. The deployment is automated through Azure Pipeline.

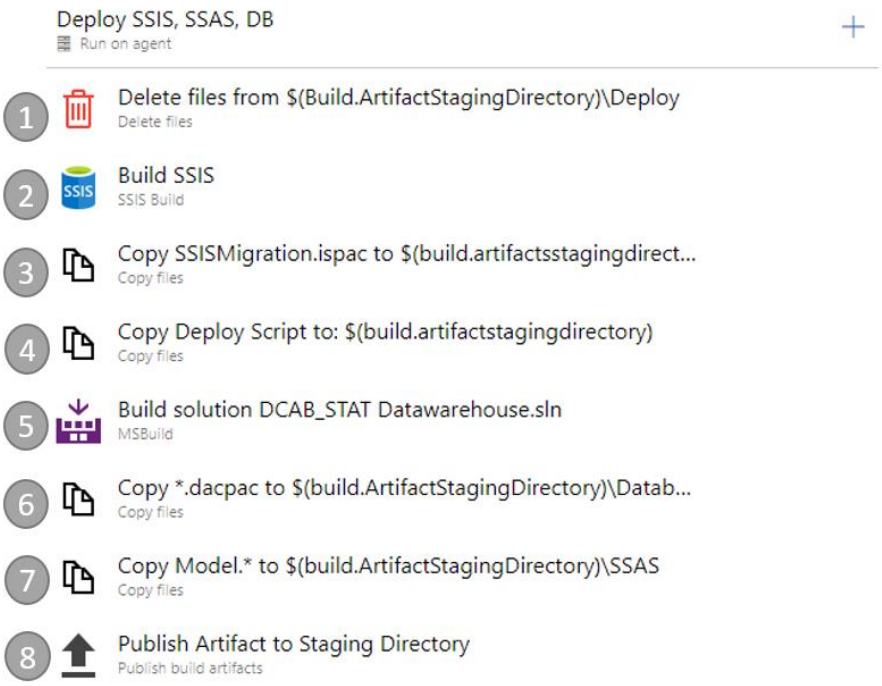



Figure 7 Deployment Pipeline in Azure Devops

The steps are as follows:

Step 1: Clean the old files from the deploy file path

Step 2: Building SSIS project

Step 3: Copy SSIS packages to SSIS folder

Step 4: Create a copy of the deploy script

Step 5: Building the datawarehouse solution

Step 6: Copy DACPAC files to Database folder

Step 7: Copy model files to SSAS folder

Step 8: The artifacts are publiched to the staging directory

Below is the path to the staging directory.

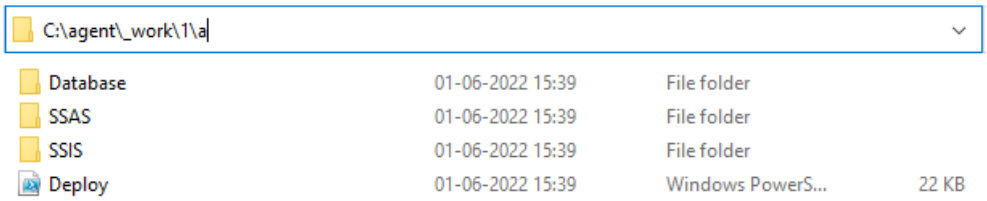
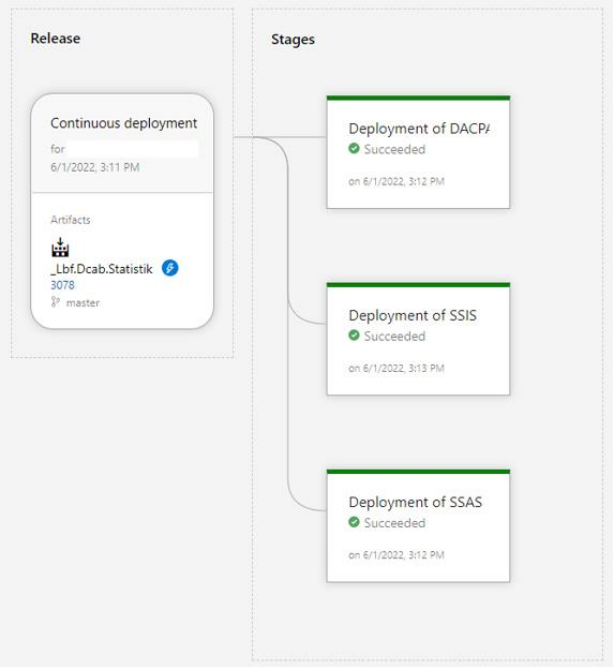



Figure 8 Staging directory

The deployment script will run different functions, deploying either the database, SSIS or SSAS, or everything, depending on which argument is set to the $DeploymentType. In figure 10 everything has been deployed successfully.




Figur 9 Deployment of DACPAC, SSIS and SSAS

When the solution is deployed, the release directory will be updated. With every deploy there is created a deployment file. If there are more than one deployment a day, the old deployment file from that day will be overwritten by the new one, but DevOps will still log the data.

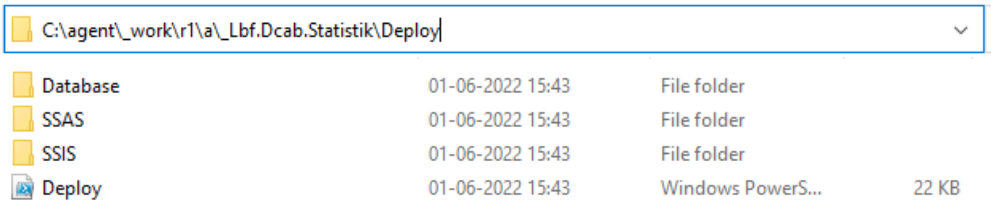



Figure 10 Release directory

# 6 Security perspective

*Do you have considerations as to what type of data you will store (GDPR), who should be allowed to see the data, and how are people limited from seeing data they should not. Do you use active directory etc. and windows authentication, or do you specify usernames and passwords.*