



# Financial Data Warehouse

- A case study

By : Kulamani Pradhan



# Agenda

- About Me
- The Challenge
- High level Solution
- Solution Details
  - Assumptions
  - Staging Layer
  - Dimensional Model
  - Control Model
  - Job Process Model
  - DGS Report for DNB
- Demo
- Questions
- Future Improvements
- Code base links



# About me

- ▶ Kulamani Pradhan
  - ▶ Native from India
  - ▶ Bachelor of Engineering in Information Technology
  - ▶ Professional experience > 9 years
  - ▶ DWH Developer
  - ▶ Hobbies include playing cricket, cooking.

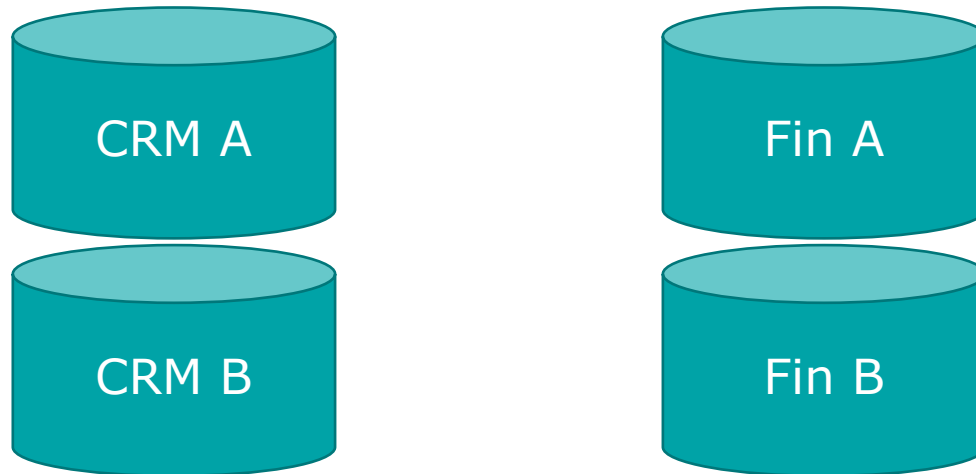
# Case

## Situation

Don is the COO at Knab. He is responsible for complying with DNB regulations, one of which is around the DGS (Deposito Garantie Stelsel). The DNB guarantees a maximum amount of € 100K per person in case the bank goes bankrupt.

In order for the DNB to make good on that promise, they want to know exactly how much money they would need to pay out. So Don wants to give them a list of all customers and the amount of money that is guaranteed by the DNB for those customers at the end of last year.

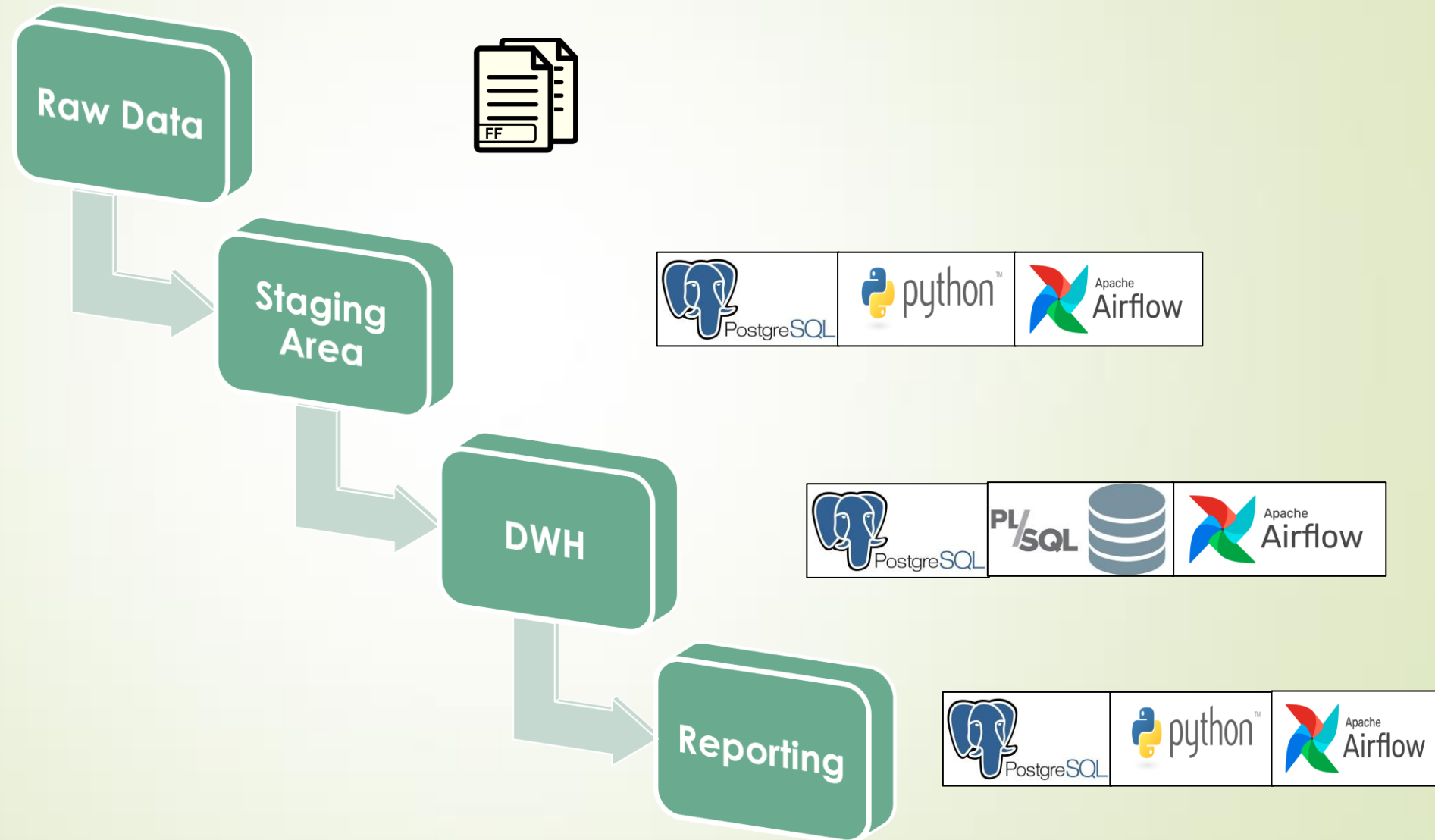
What makes it slightly more interesting, there are multiple source databases with information about customers (CRM) and about financial transactions (let's call them Fin) and customers can appear in multiple systems at once.



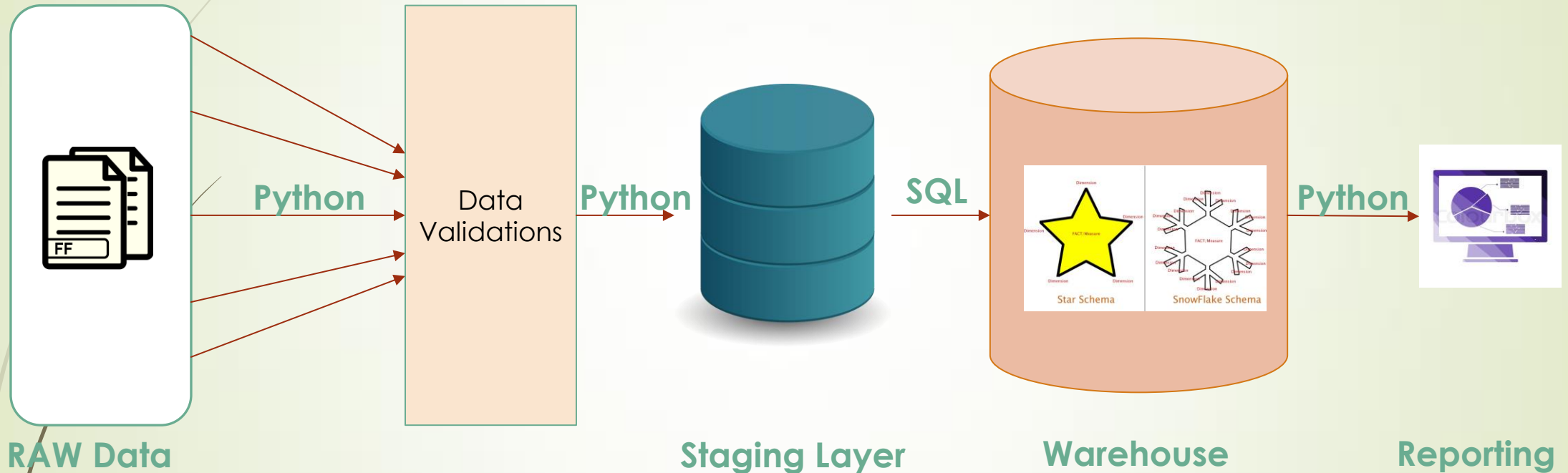
## Challenges

- Describe which data you think should be used to answer Don's question
- Draw a dimensional data model describing these data sets
- Describe which controls you would implement to ensure the final result is correct.
- Write a SQL query that produces the desired result

# High Level Solution : Approach



# High Level Solution : Approach

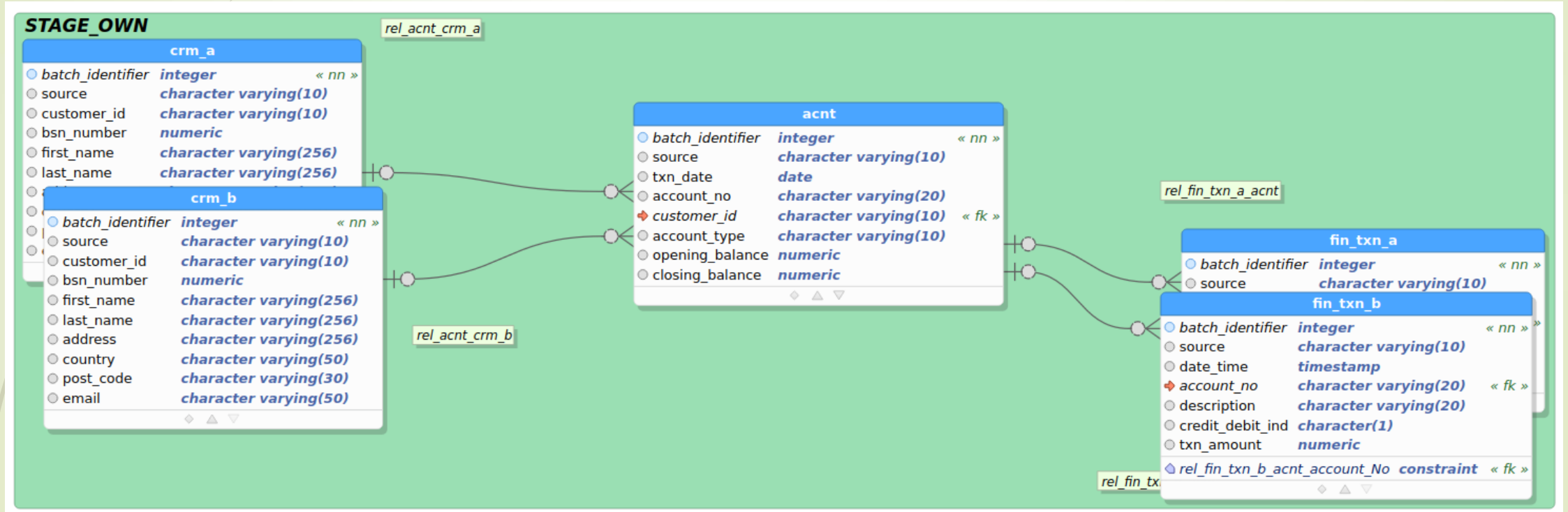




# Assumptions

- CRM system is an internal OLTP system of Knab. If we have external files then approach will change.
- A financial transaction is related to an account and that account must belong to a customer. A customer can have multiple accounts on their name.
- An account can be allocated to multiple customers
- We will receive an acknowledgement file with filename & record count of each and every source files.
- We will receive an account file with opening balance and closing balance in it on daily basis.
- DNB is giving guarantee 100K per customer, doesn't matter how many accounts that customer has.
- All attributes within staging table.
- Flat file are in csv format and both crm\_a & crm\_b has similar structure. Same for fin\_txn\_a & fin\_txn\_b.

# Solution Details : Staging Model



Staging Loading Job :

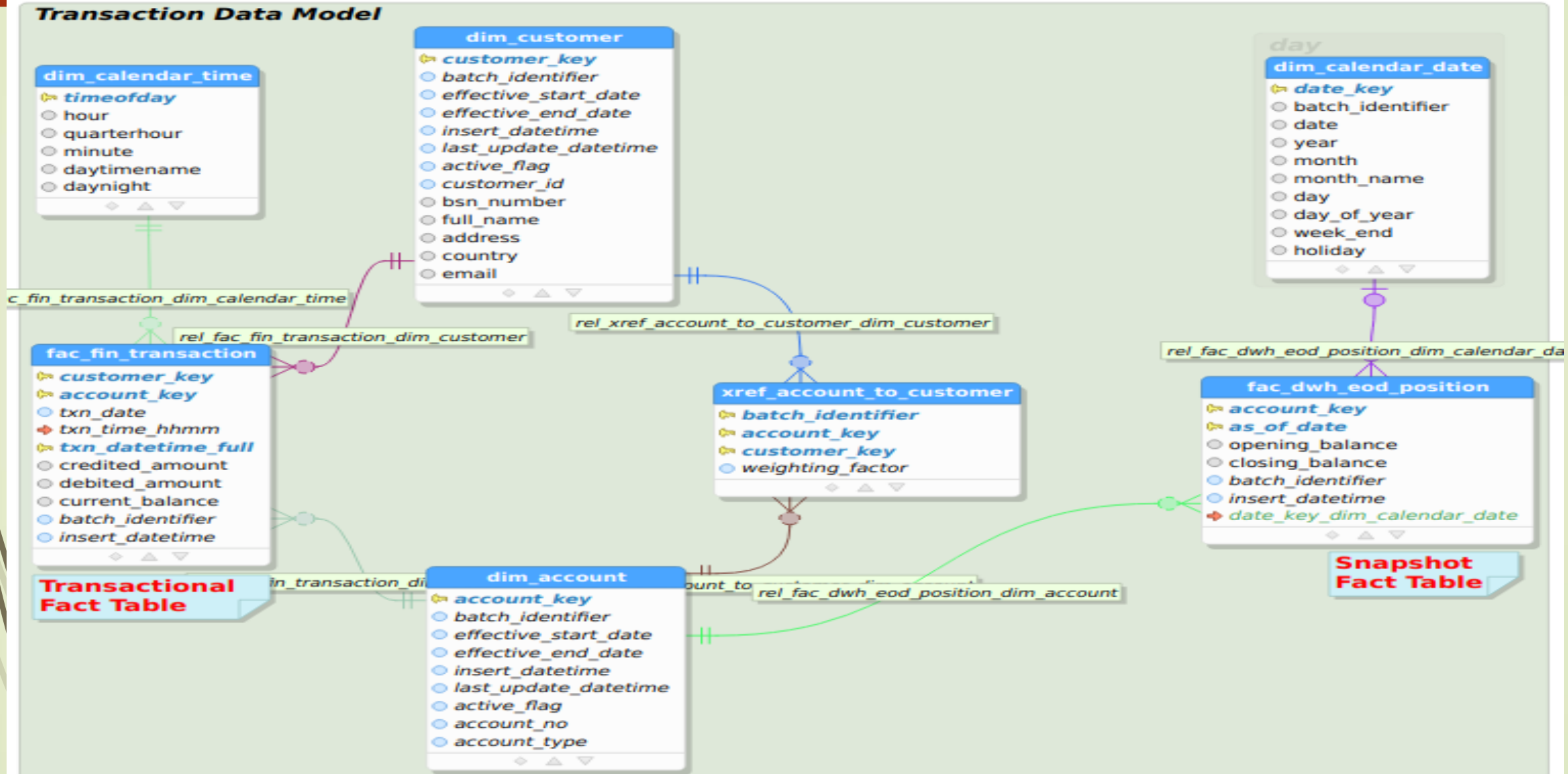
[https://github.com/Kulamanipradhan0/knab\\_dwh/blob/main/etl\\_repo/staging/stg\\_load.py](https://github.com/Kulamanipradhan0/knab_dwh/blob/main/etl_repo/staging/stg_load.py)



# Solution Details : Data Validations

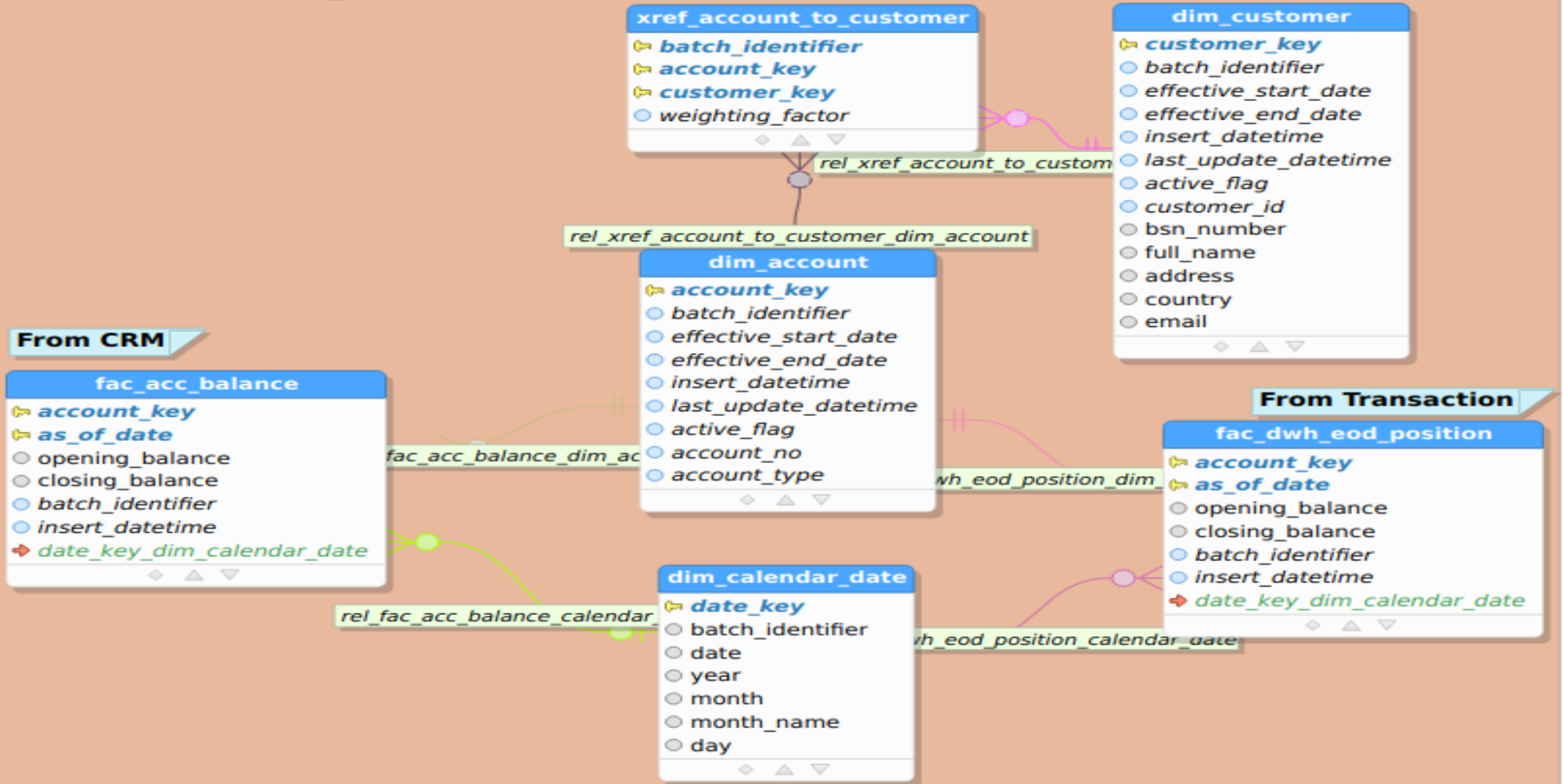
- Data Completeness Validation
  - knab\_dwh\_ack.txt file containing (file\_name, record\_count)
- Data Correctness Validation
  - *Mandatory Constraints* (customer\_id , account\_no etc)
  - *Data-Type Constraints* (Txn Amount must be Numeric)
  - *Duplicate Record Constraints*
  - *Attribute Value level Constraints* (customer\_id must starts with "CA")
  - *Date/Time Format Constraints* (Ex : yyyy-mm-dd)
  - *Referential Integrity Constraints*
- Storing bad records in error\_log table to monitor and discuss with source systems to fix if required.

# Solution Details :Dimensional Model



# Solution Details :Control Model

## Control Model(dw\_h\_ow)



# Solution Details : Report

## CRM- A

customer_id	bsn_number	first_name	last_name
CA7405	20000006	Simona	Morasca
CA7406	20000007	Mitsue	Tollner
CA7010	20000011	Minna	Amigon

## CRM- B

customer_id	bsn_number	first_name	last_name
CA7401	20000001	James	Butt
CA7404	20000005	Donette	Foller
CA7010	20000011	Minna	Amigon

## ACNT

txn_date	account no	customer id	account_type	opening_balance	closing_balance
2020/12/31 00:00:00.000	KN110000000001	CA7401	Current	0	500
2020/12/31 00:00:00.000	KN110000000002	CA7401	Saving	46000	45500
2020/12/31 00:00:00.000	KN110000000003	CA7401	DMAT	56000	56000
2020/12/31 00:00:00.000	KN110000000004	CA7010	Current	1000	1200
2020/12/31 00:00:00.000	KN110000000005	CA7404	Current	950	900
2020/12/31 00:00:00.000	KN110000000006	CA7405	Current	155000	254900
2020/12/31 00:00:00.000	KN110000000006	CA7406	Current	155000	254900

# Solution Details : Report

## KNAB - DNB Deposito Garantie Stelsel Report

Report Generation Time : 12/02/2023 18:33:53

Total Customer : 5

Total Closing Balance : 359000.0


Total Pay Out Amount : 302100.0

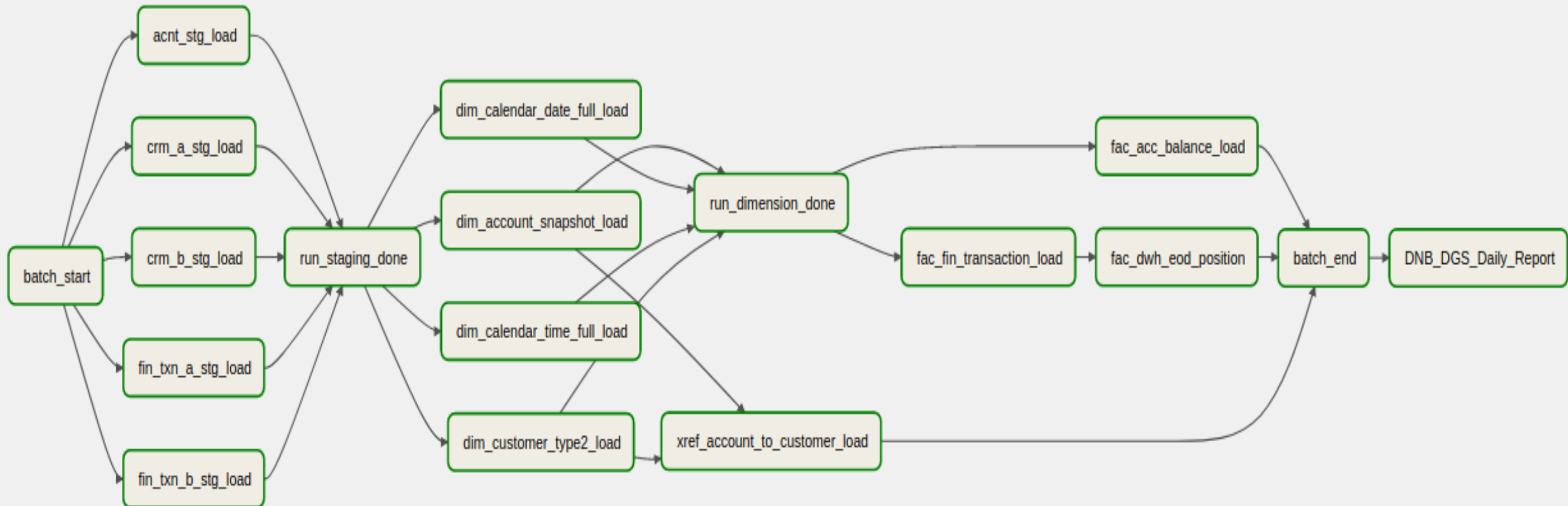
Year	Social Security Number	Customer name	Closing Balance	Pay Out Amount	Currency
2020	20000001	James Butt	102000	100000	EUR
2020	20000005	Donette Foller	900	900	EUR
2020	20000006	Simona Morasca	127450	100000	EUR
2020	20000007	Mitsue Tollner	127450	100000	EUR
2020	20000011	Minna Amigon	1200	1200	EUR

# Solution Details : Job Process Model

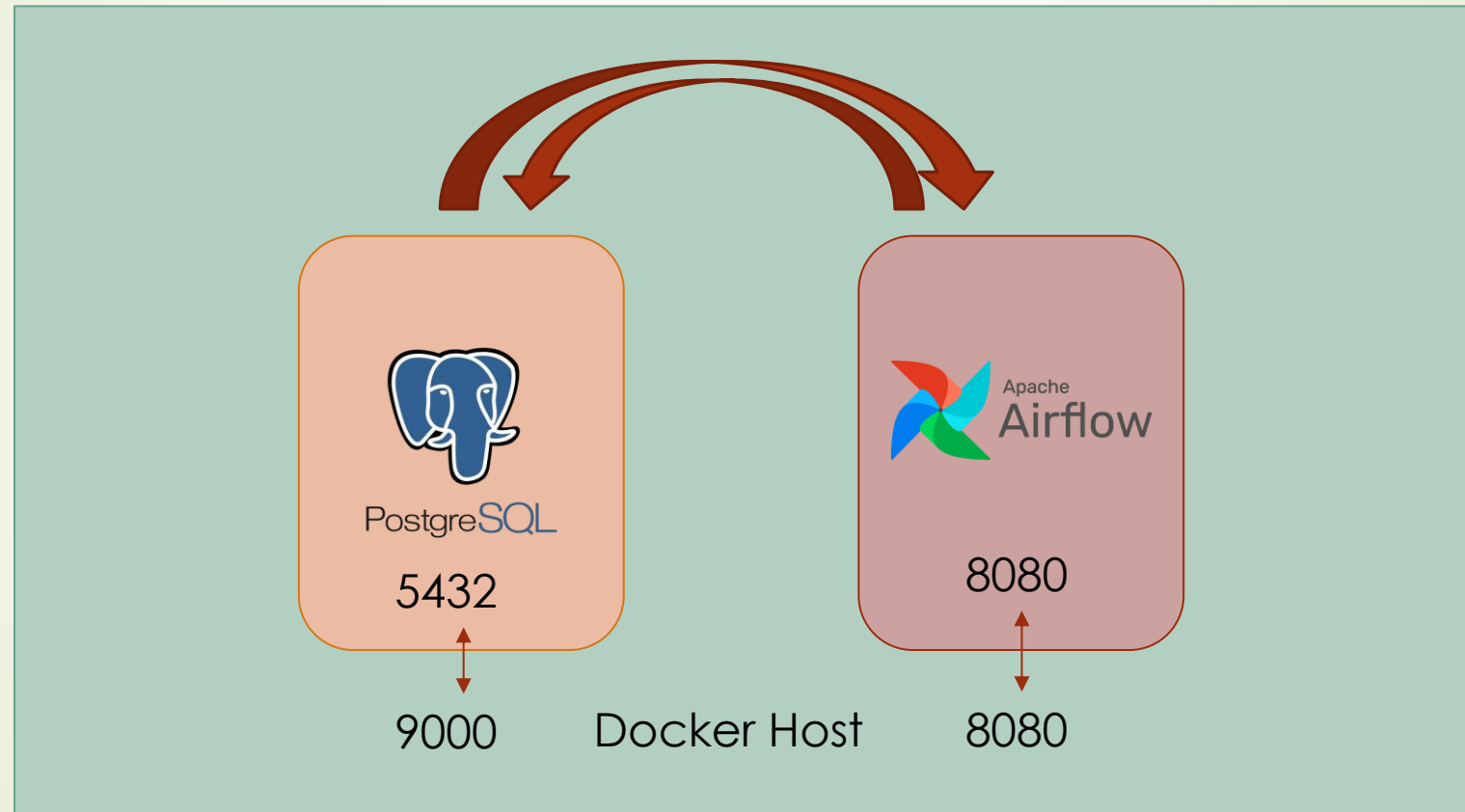
BashOperator

queued running success failed up\_for\_retry up\_for\_reschedule upstream\_failed skipped scheduled no\_status

Auto-refresh 



# Containerization :



Container App	Docker File name	Image name(Docker hub)
PostgreSQL	Dockerfile_pg	mekpradhan/knab_dwh:knab_dwh_pg
Airflow	Dockerfile_airflow	mekpradhan/knab_dwh:Knab_dwh_airflow



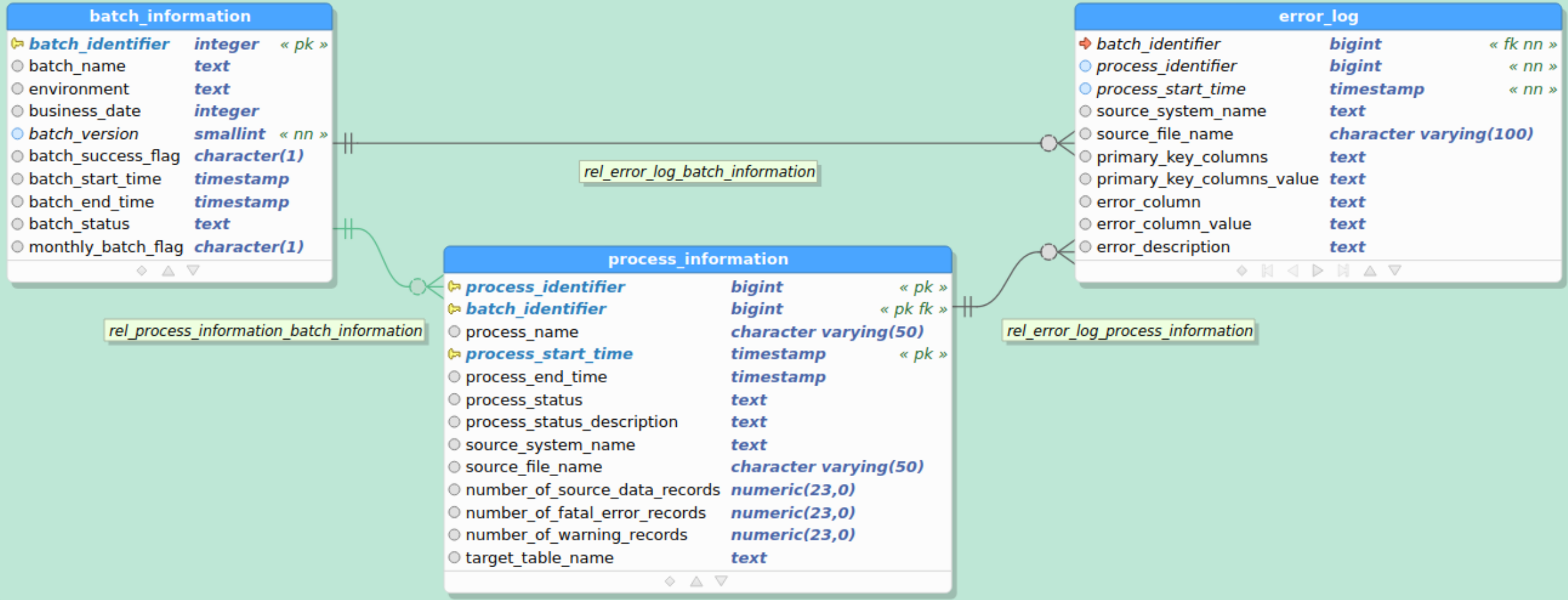
# Demo

- Done on a small set of Data(*in the interest of time*)
- Will be an E2E demo inclusive of:
  - SQL code walkthrough
  - Job execution using Airflow



# Solution Details : Data Model(Auditing)

## AUDITING\_OWN



Questions?



# Future Improvements

- Event Based Trigger : (Automatic triggering of DWH batch)
  - We can create an airflow dag task to check every 2-5 seconds, if already all files have been delivered then start the batch.
- Batch Status Mail :
  - We can create a job which will send (Team) an email confirmation that batch has started, completed staging, completed fact and OLAP etc. In each stage we can get notified.
- Staging Area :
  - Parallel processing : If we use Amazon redshift(distributed network) to store whole DWH area, then these huge data can be processed parallel across multiple nodes.
- Batch Monitoring Dashboard :
  - Create a Batch monitoring dashboard using auditing\_own schema tables, where we can easily monitor the batch status(if not airflow).
- Source File Movement :
  - Once Python processes the source file, the file should be moved to done location. In that way we will not be overriding the files of one date to another.
  - In case there is a failure we should move the file to working location. In that way it would be clear which one is still pending for processing.



# Future Improvements

- Independent Execution of airflow containers :
  - In a production environment set up , we should separate airflow scheduler & webserver, so that they should run in two different containers.
  - Proper networking between containers.
  - Creating an docker-compose.yml file to start all these containers at once
- PDF Report to DNB

# Code Base Links :

- Docker Images(Airflow + Postgres)
  - Dockerfiles : [https://github.com/Kulamanipradhan0/knab\\_dwh](https://github.com/Kulamanipradhan0/knab_dwh)
    - Dockerfile\_pg
    - Dockerfile\_airflow
  - Images : [https://hub.docker.com/repository/docker/mekpradhan/knab\\_dwh](https://hub.docker.com/repository/docker/mekpradhan/knab_dwh)
- DDL Code base : This is used to create all DB objects i.e. schema, users, tables, partitions etc
  - [https://github.com/Kulamanipradhan0/knab\\_dwh/tree/main/db\\_scripts](https://github.com/Kulamanipradhan0/knab_dwh/tree/main/db_scripts)
- ETL Repository : All code base related to staging, dimension, fact & report loading
  - [https://github.com/Kulamanipradhan0/knab\\_dwh/tree/main/etl\\_repo](https://github.com/Kulamanipradhan0/knab_dwh/tree/main/etl_repo)
- Sample Source Files : All sample flat files
  - [https://github.com/Kulamanipradhan0/knab\\_dwh/tree/main/src\\_files/polling](https://github.com/Kulamanipradhan0/knab_dwh/tree/main/src_files/polling)
- Airflow Dag file : Used for running the complete DWH batch
  - [https://github.com/Kulamanipradhan0/knab\\_dwh/tree/main/airflow/dags](https://github.com/Kulamanipradhan0/knab_dwh/tree/main/airflow/dags)

# Code Base Links :

- Loading data from Transaction to EOD Position : (Making sure all active customers are getting loaded everyday )
  - [https://github.com/Kulamanipradhan0/knab\\_dwh/blob/main/etl\\_repo/fact/fac\\_dwh\\_eod\\_position.sql](https://github.com/Kulamanipradhan0/knab_dwh/blob/main/etl_repo/fact/fac_dwh_eod_position.sql)
- Data Validation Layer(nice to have). Ex:
  - [https://github.com/Kulamanipradhan0/knab\\_dwh/blob/main/etl\\_repo/staging/staging\\_data\\_validation.sql](https://github.com/Kulamanipradhan0/knab_dwh/blob/main/etl_repo/staging/staging_data_validation.sql)
- ETL Log files : During Job run ETL logs will be created in this directory.
  - [https://github.com/Kulamanipradhan0/knab\\_dwh/tree/main/log\\_files](https://github.com/Kulamanipradhan0/knab_dwh/tree/main/log_files)
- SCD Type 2 Implementation using SQL : (dim\_customer)
  - [https://github.com/Kulamanipradhan0/knab\\_dwh/blob/main/etl\\_repo/dimension/dim\\_customer\\_type2\\_load.sql](https://github.com/Kulamanipradhan0/knab_dwh/blob/main/etl_repo/dimension/dim_customer_type2_load.sql)
- Report Code : (DNB\_DGS\_Last\_Year\_Report)
  - Main Job :  
[https://github.com/Kulamanipradhan0/knab\\_dwh/blob/main/etl\\_repo/report/DNB\\_DGS\\_Last\\_Year\\_Report.py](https://github.com/Kulamanipradhan0/knab_dwh/blob/main/etl_repo/report/DNB_DGS_Last_Year_Report.py)
  - SQL :  
[https://github.com/Kulamanipradhan0/knab\\_dwh/blob/main/etl\\_repo/report/dnb\\_dgs\\_report\\_query.sql](https://github.com/Kulamanipradhan0/knab_dwh/blob/main/etl_repo/report/dnb_dgs_report_query.sql)



