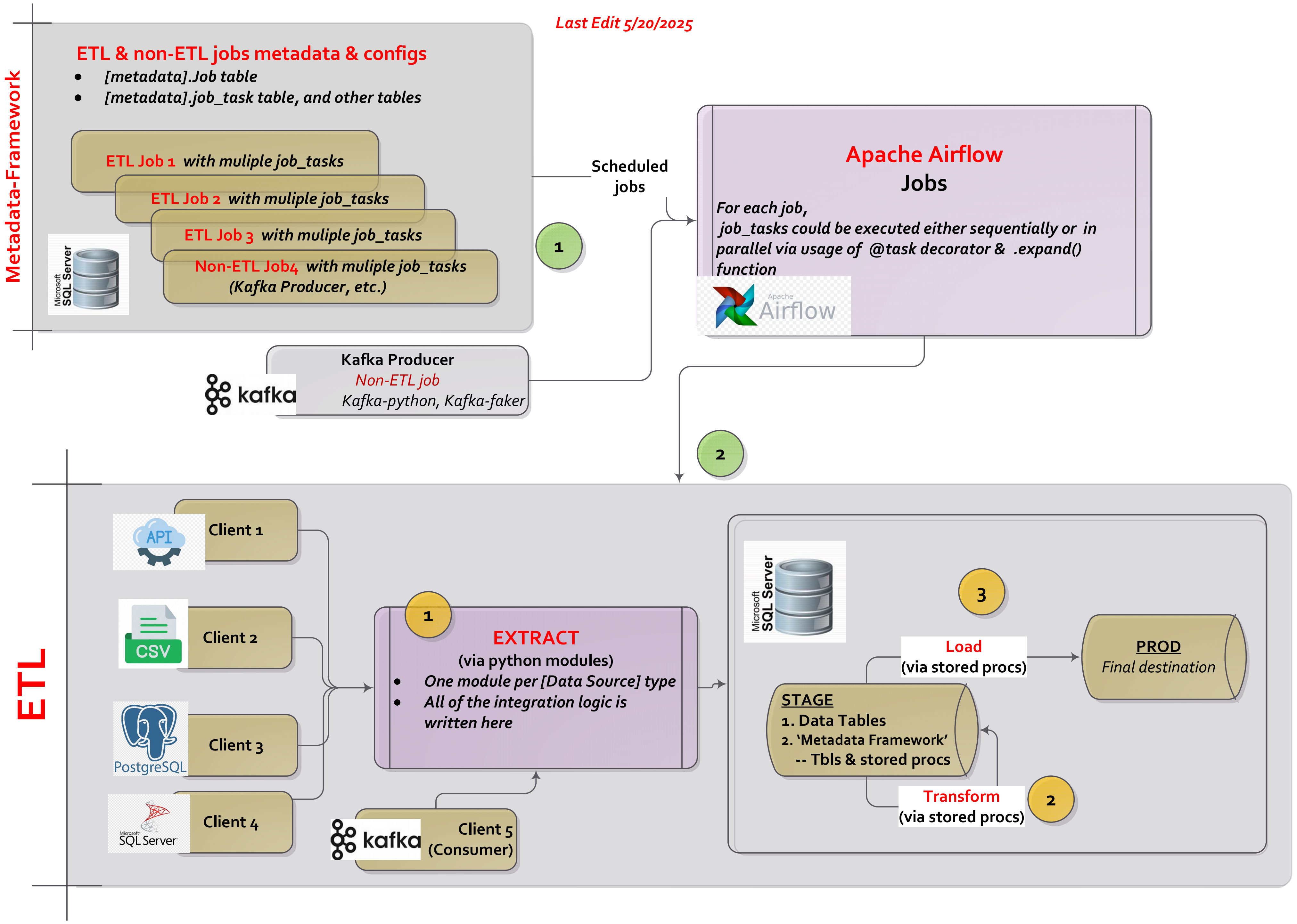
ETL Pipelines with Apache Airflow – System Architecture

(*Integrating multiple Data-Sources & a custom Metadata-Framework*)

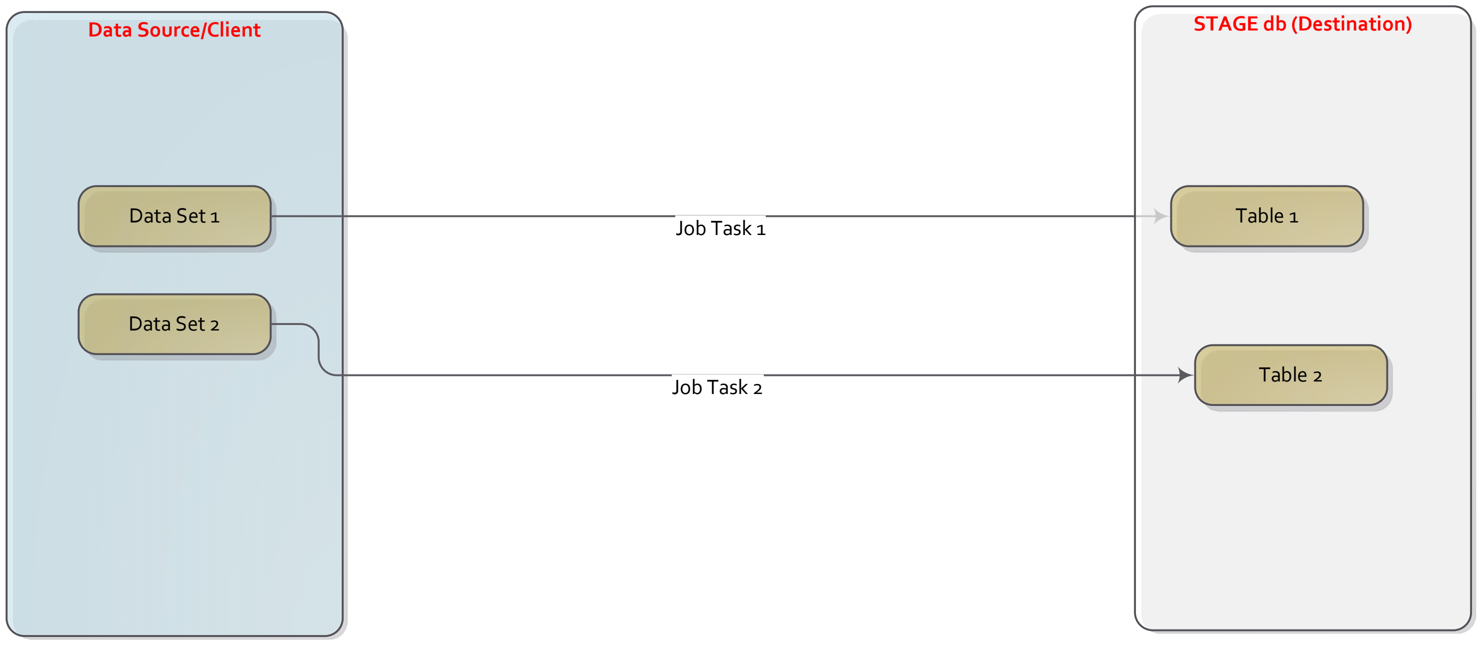
Last Edit 5/20/2025



Metadata-Framework
ETL & non-ETL jobs metadata & configs
• [metadata].Job table
• [metadata].job_task table, and other tables
ETL Job 1 with muliple job_tasks
ETL Job 2 with muliple job_tasks
ETL Job 3 with muliple job_tasks
Non-ETL Job4 with muliple job_tasks (Kafka Producer, etc.)
Microsoft SQL Server
1
Scheduled jobs
Apache Airflow
Jobs
For each job, job_tasks could be executed either sequentially or in parallel via usage of @task decorator & .expand() function
Apache Airflow
Kafka Producer
Non-ETL job
Kafka-python, Kafka-faker
kafka
2
ETL
API
Client 1
CSV
Client 2
PostgreSQL
Client 3
Microsoft SQL Server
Client 4
1
EXTRACT
(via python modules)
• One module per [Data Source] type
• All of the integration logic is written here
kafka
Client 5 (Consumer)
Microsoft SQL Server
3
Load (via stored procs)
PROD
Final destination
STAGE
1. Data Tables
2. 'Metadata Framework' -- Tbls & stored procs
Transform (via stored procs)
2

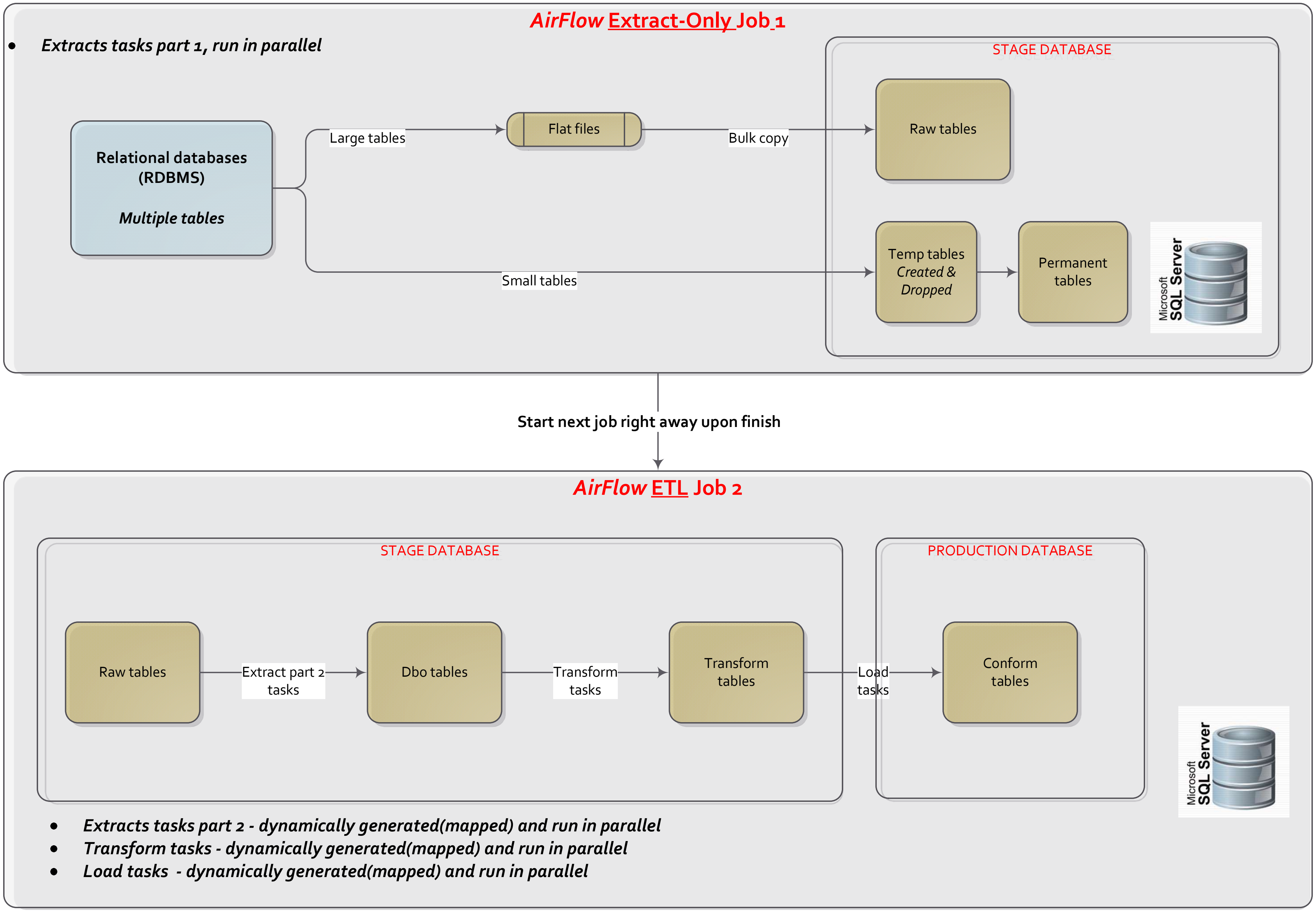
The typical logical structure of the ETL job (Apache Airflow DAG)

- A Single Job per Data Source/Client
- Multiple Job Tasks per Job.
- Each Task corresponds to a single Destination table



Workflow for bringing data based on its size (large/small) via two consecutive jobs. Data source – RDBMS.

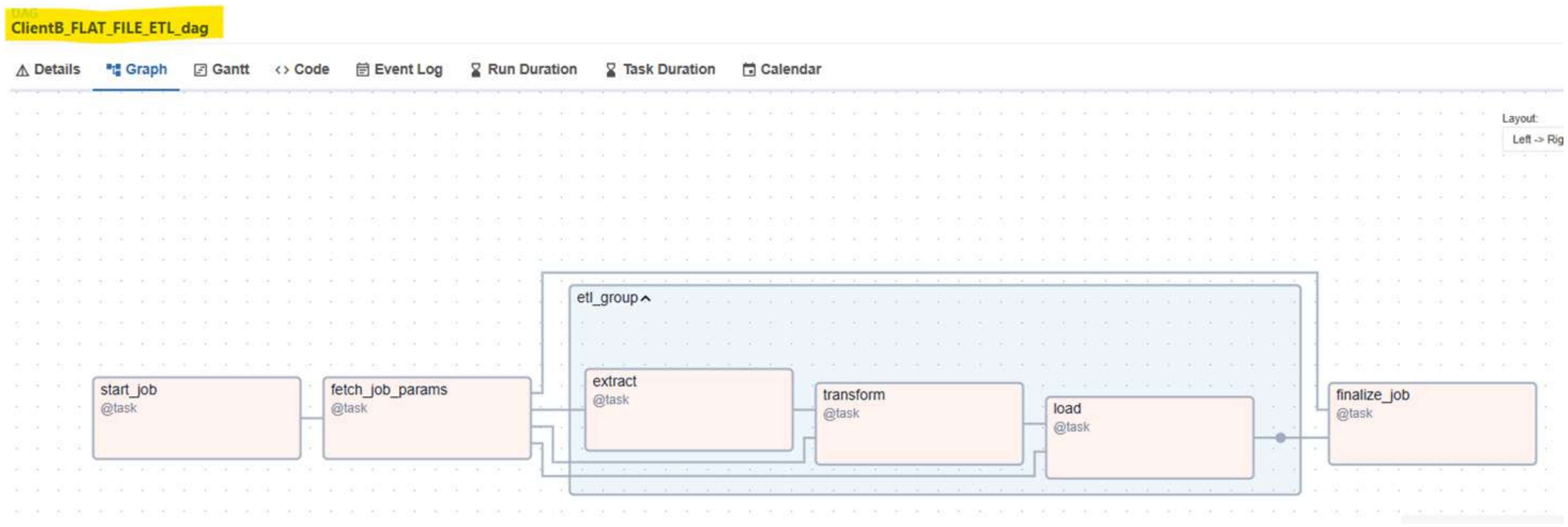
Last Edit 5/20/2025


AirFlow Extract-Only Job 1
Extracts tasks part 1, run in parallel
STAGE DATABASE
Relational databases (RDBMS)
Multiple tables
Large tables
Flat files
Bulk copy
Raw tables
Small tables
Temp tables Created & Dropped
Permanent tables
Microsoft SQL Server
Start next job right away upon finish
AirFlow ETL Job 2
STAGE DATABASE
PRODUCTION DATABASE
Raw tables
Extract part 2 tasks
Dbo tables
Transform tasks
Transform tables
Load tasks
Conform tables
Microsoft SQL Server
Extracts tasks part 2 - dynamically generated(mapped) and run in parallel
Transform tasks - dynamically generated(mapped) and run in parallel
Load tasks - dynamically generated(mapped) and run in parallel

Simple ETL job in Apache Airflow

The tasks within Extract, Transform, Load phases are processed sequentially.

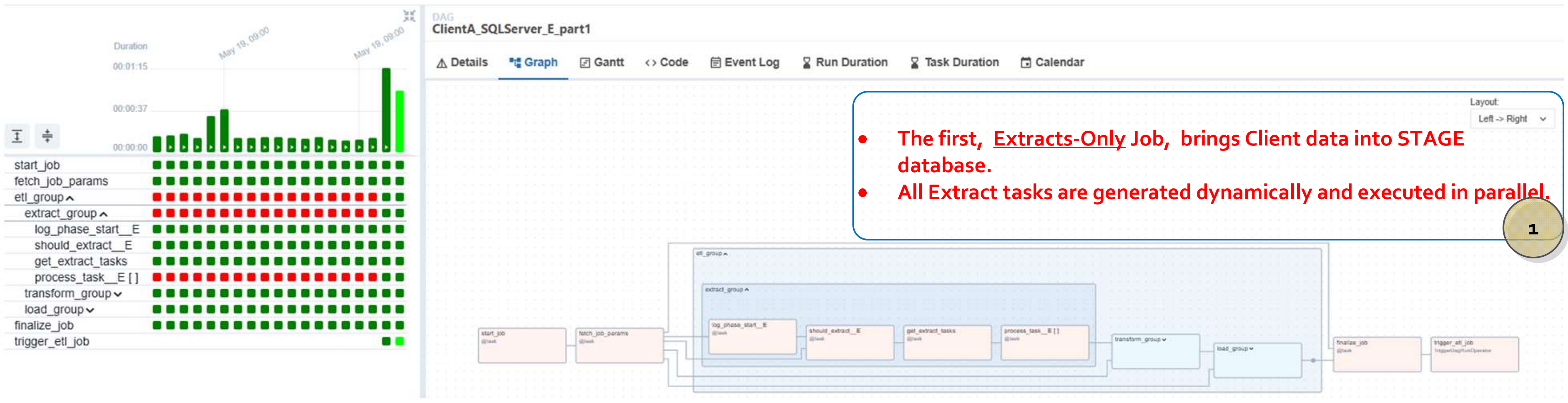
Last Edit 5/20/2025



- **This ETL job processes data sequentially.**

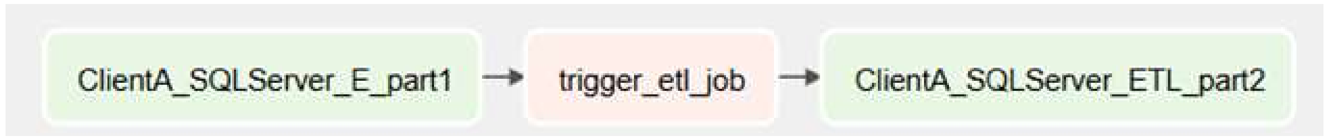
Example of a complex DAG - Processing ETL tasks in parallel via two jobs.

Last Edit 5/20/2025



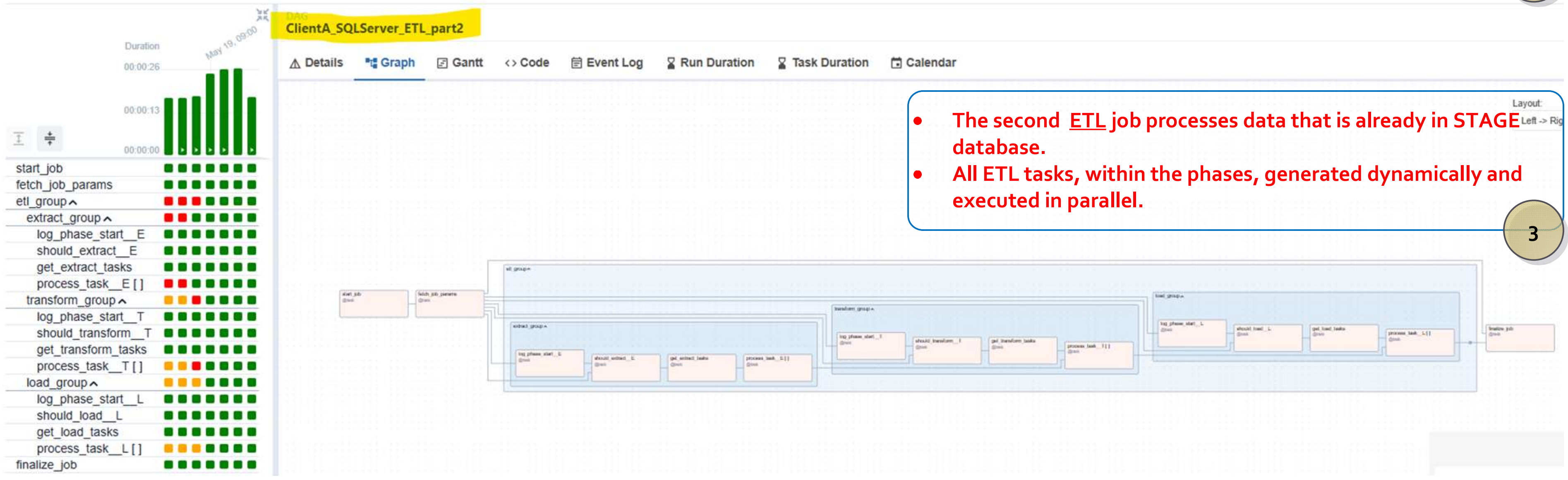
- The first, Extracts-Only Job, brings Client data into STAGE database.
- All Extract tasks are generated dynamically and executed in parallel.

1



Once the first job is done, the second ETL job is triggered.

2



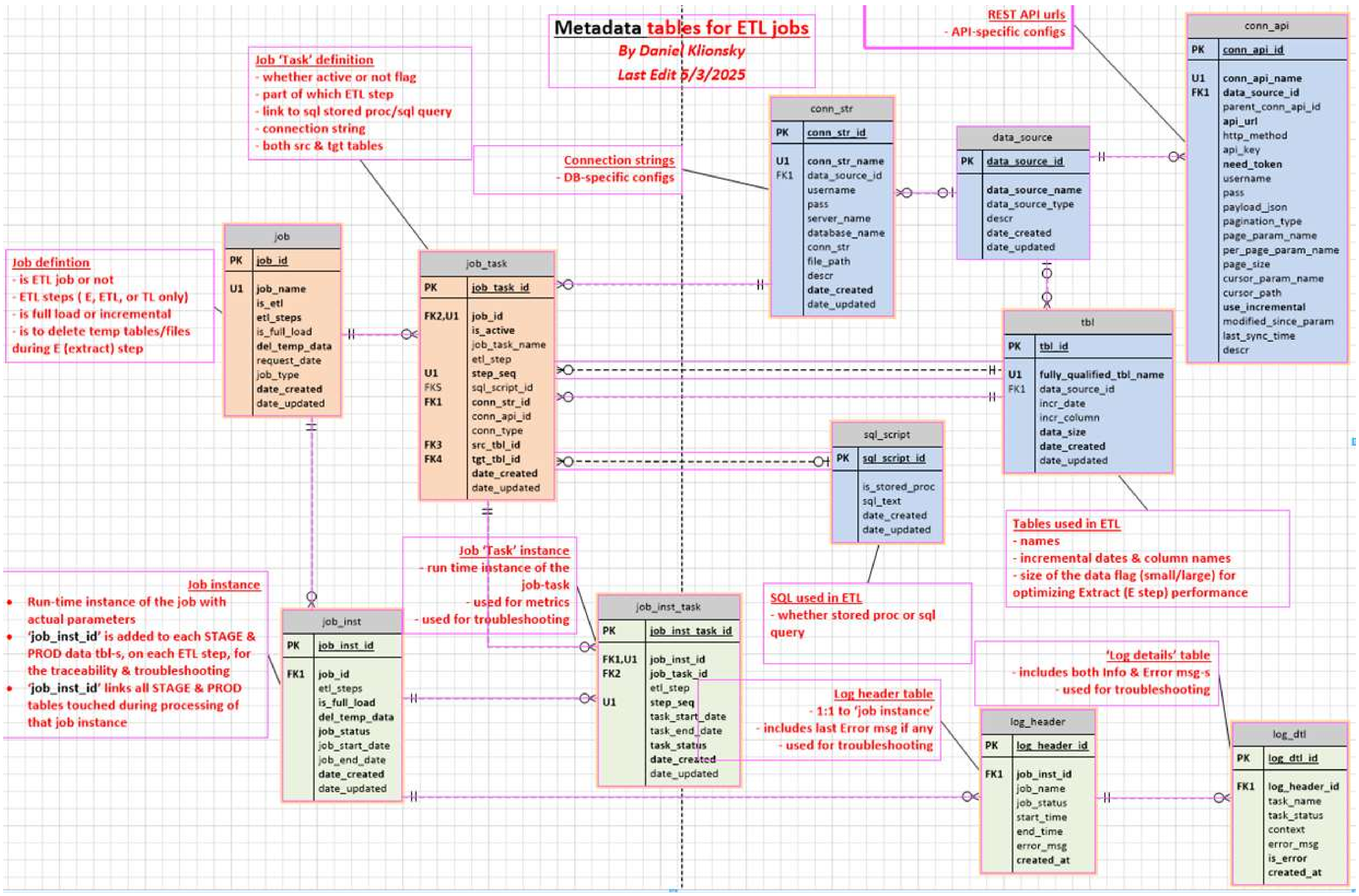
- The second ETL job processes data that is already in STAGE database.
- All ETL tasks, within the phases, generated dynamically and executed in parallel.

3

Metadata tables for ETL jobs

By Daniel Klionsky

Last Edit 5/3/2025


REST API urls
- API-specific configs
Job 'Task' definition
- whether active or not flag
- part of which ETL step
- link to sql stored proc/sql query
- connection string
- both src & tgt tables
Connection strings
- DB-specific configs
Job defintion
- is ETL job or not
- ETL steps (E, ETL, or TL only)
- is full load or incremental
- is to delete temp tables/files during E (extract) step
job
PK job_id
U1 job_name
is_etl
etl_steps
is_full_load
del_temp_data
request_date
job_type
date_created
date_updated
job_task
PK job_task_id
FK2,U1 job_id
is_active
job_task_name
etl_step
U1 step_seq
FK5 sql_script_id
FK1 conn_str_id
conn_api_id
conn_type
FK3 src_tbl_id
FK4 tgt_tbl_id
date_created
date_updated
conn_str
PK conn_str_id
U1 conn_str_name
FK1 data_source_id
username
pass
server_name
database_name
conn_str
file_path
descr
date_created
date_updated
data_source
PK data_source_id
data_source_name
data_source_type
descr
date_created
date_updated
conn_api
PK conn_api_id
U1 conn_api_name
FK1 data_source_id
parent_conn_api_id
api_url
http_method
api_key
need_token
username
pass
payload_json
pagination_type
page_param_name
per_page_param_name
page_size
cursor_param_name
cursor_path
use_incremental
modified_since_param
last_sync_time
descr
tbl
PK tbl_id
U1 fully_qualified_tbl_name
FK1 data_source_id
incr_date
incr_column
data_size
date_created
date_updated
sql_script
PK sql_script_id
is_stored_proc
sql_text
date_created
date_updated
Tables used in ETL
- names
- incremental dates & column names
- size of the data flag (small/large) for optimizing Extract (E step) performance
Job 'Task' instance
- run time instance of the job-task
- used for metrics
- used for troubleshooting
Job instance
• Run-time instance of the job with actual parameters
• 'job_inst_id' is added to each STAGE & PROD data tbl-s, on each ETL step, for the traceability & troubleshooting
• 'job_inst_id' links all STAGE & PROD tables touched during processing of that job instance
job_inst
PK job_inst_id
FK1 job_id
etl_steps
is_full_load
del_temp_data
job_status
job_start_date
job_end_date
date_created
date_updated
job_inst_task
PK job_inst_task_id
FK1,U1 job_inst_id
FK2 job_task_id
etl_step
U1 step_seq
task_start_date
task_end_date
task_status
date_created
date_updated
SQL used in ETL
- whether stored proc or sql query
Log header table
- 1:1 to 'job instance'
- includes last Error msg if any
- used for troubleshooting
'Log details' table
- includes both Info & Error msg-s
- used for troubleshooting
log_header
PK log_header_id
FK1 job_inst_id
job_name
job_status
start_time
end_time
error_msg
created_at
log_dtl
PK log_dtl_id
FK1 log_header_id
task_name
task_status
context
error_msg
is_error
created_at