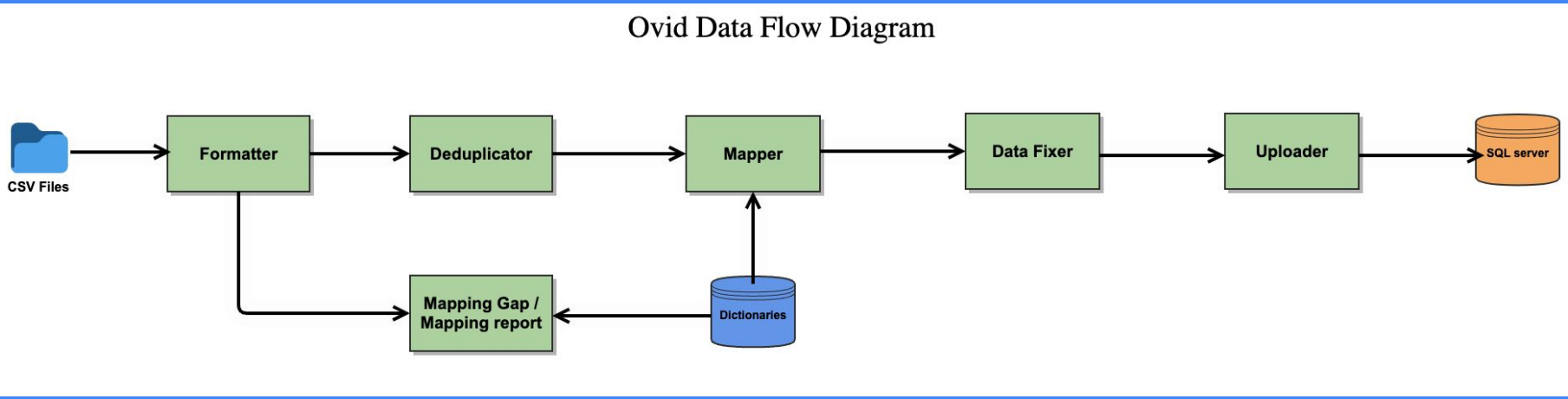
Ovid: OMOP to PCORnet converter

Presented by: Ali Nouina



- 6 main components
- Can all be run at once, combination, or individually
- Runs on the pyspark cluster

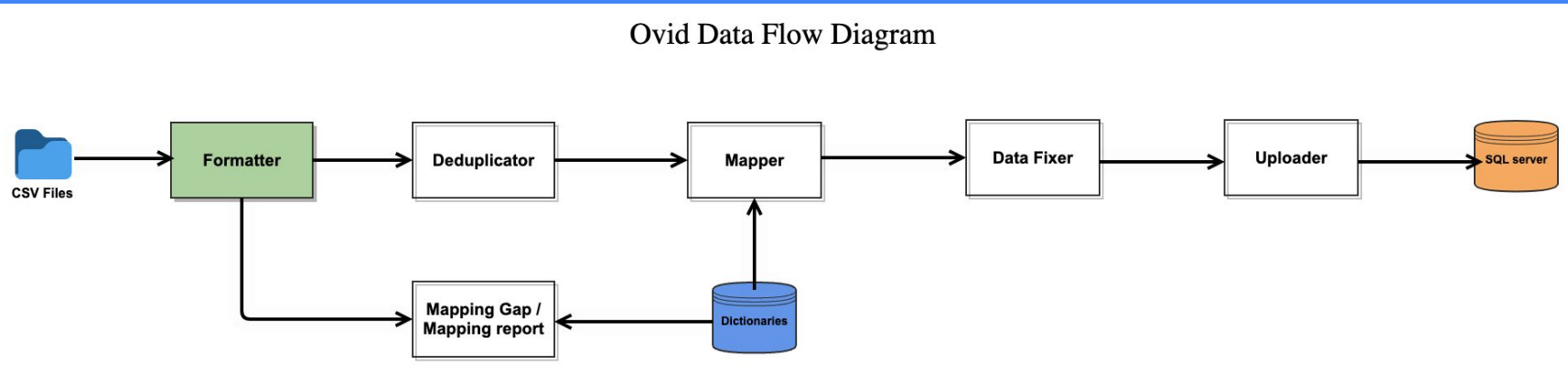
onefl_converter.py

common/
----> commonFunctions.py
----> spark_secrets.py
----> settings.py

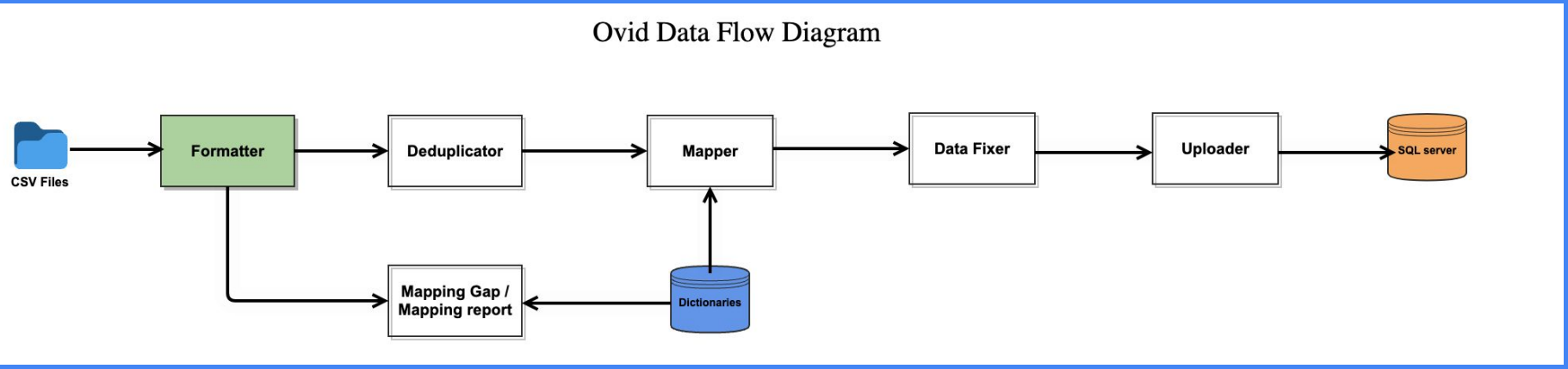
mapping_scripts/
----> demographic_mapper.py
----> encounter_mapper.py
----> .....
----> ....

partners/
----> partner_1
----------> dictionaries.py
----------> formatter_scripts/
------------------->demographic_formatter.py
------------------->encounter_formatter.py
-------------------> .....
-------------------> ....
----------> data/
-------------------> formatter_output/
-------------------> deduplicator_output
-------------------> mapper_output
-------------------> mapping_gap_output/

----> partner_2
----> ....



- Reads from the input directory
- Converts input data to a standard PCORnet format
- Each table have its own formatter
- Each partners has its own set of formatters



Parameters:

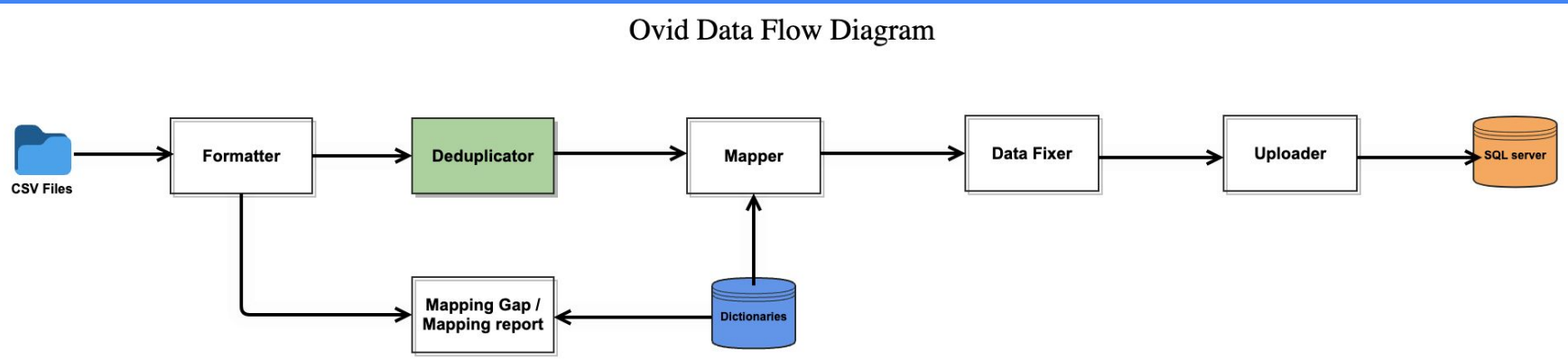
- Input directory: -d [/path/to/data/parent/folder/]
- Partner : -p [partner_name]
- Job : -j **format**
- Folder : -f [folder_name1 folder_name2 …]
- Table : -t [table_name1 table_name2 …]

Example 1: Format the demographic and encounter tables:

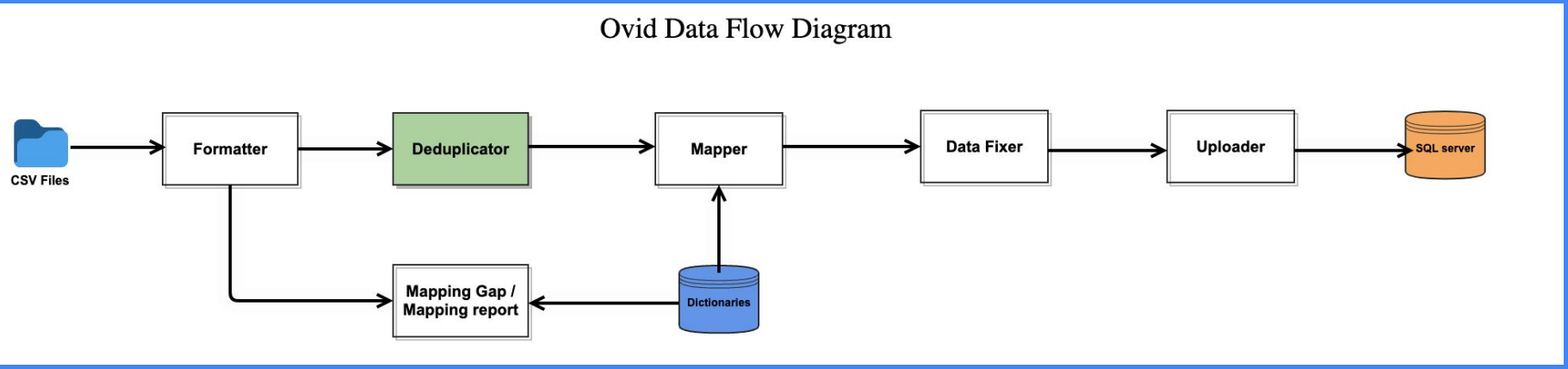
cluster run -a -d [/path/to/data/parent/folder/] -- onefl_converter.py -p partner_a -j **format** -t demographic encounter -f folder_1

Example 2: Format all the tables:

cluster run -a -d [/path/to/data/parent/folder/] -- onefl_converter.py -p partner_a -j **format** -t all -f folder_1



- Reads from the formatter output
- Removes duplicates id from the formatted data



Parameters:

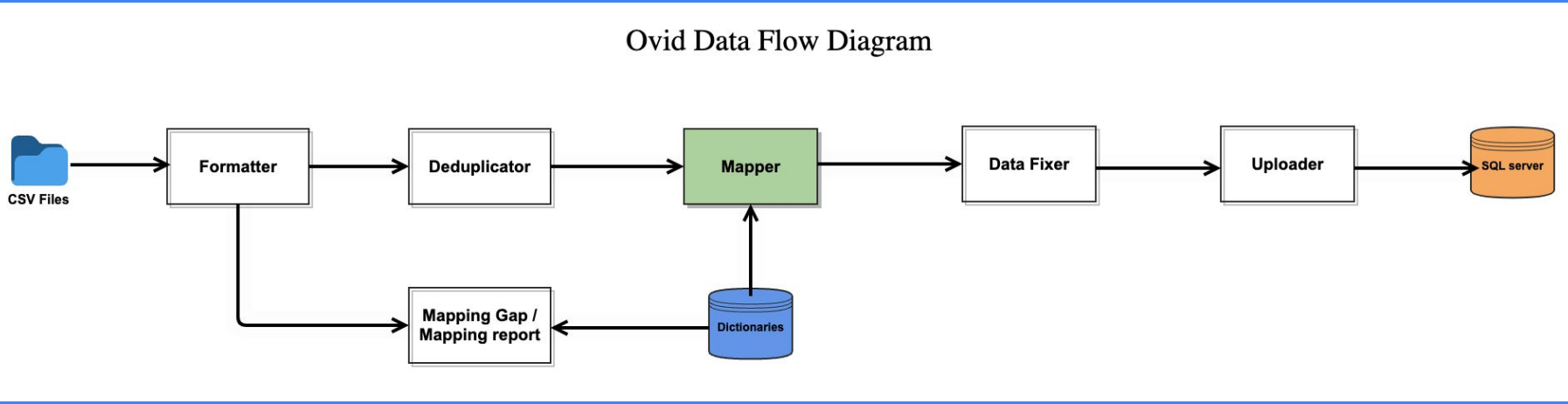
- Partner : -p [partner_name]
- Job : -j **deduplicate**
- Folder : -f [folder_name1 folder_name2 …]
- Table : -t [table_name1 table_name2 …]

Example 1: Deduplicate the demographic and encounter tables:

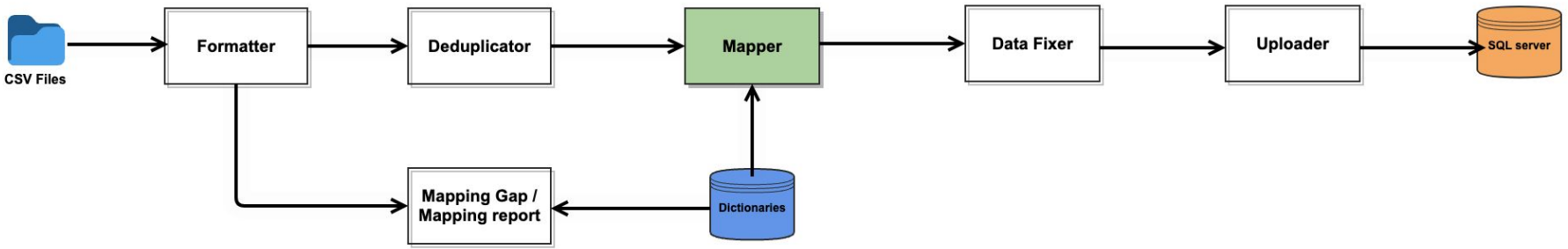
cluster run -a -- onefl_converter.py -p partner_a -j **deduplicate** -t demographic encounter -f folder_1

Example 2: deduplicate all the tables:

cluster run -a -- onefl_converter.py -p partner_a -j **deduplicate** -t all -f folder_1



- Reads from the deduplicator output
- Uses the partner custom dictionaries to map the deduplicated data



Parameters:

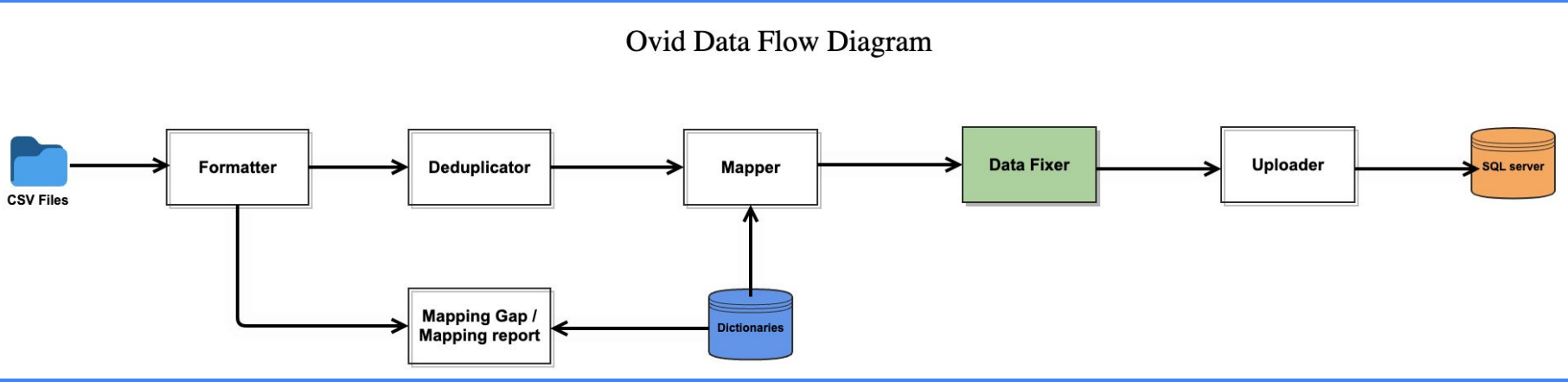
- Partner : -p [partner_name]
- Job : -j **map**
- Folder : -f [folder_name1 folder_name2 …]
- Table : -t [table_name1 table_name2 …]

Example 1: Map the demographic and encounter tables:

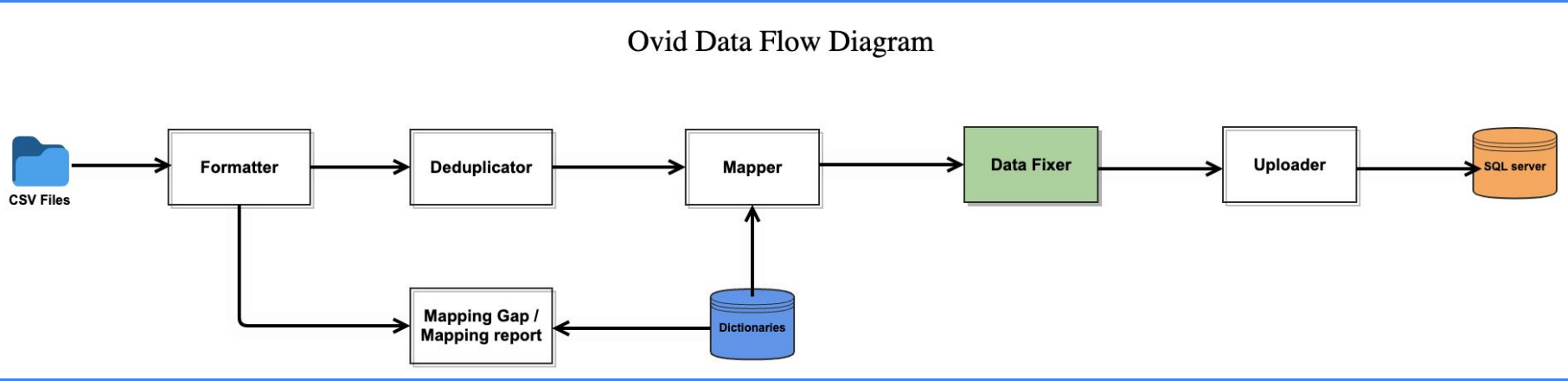
cluster run -a -- onefl_converter.py -p partner_a -j **map** -t demographic encounter -f folder_1

Example 2: Map all the tables:

cluster run -a -- onefl_converter.py -p partner_a -j **map** -t all -f folder_1



- Reads from the mapper output
- Uses the partner custom fixers to call and apply multiple common and custom data fixes



Parameters:

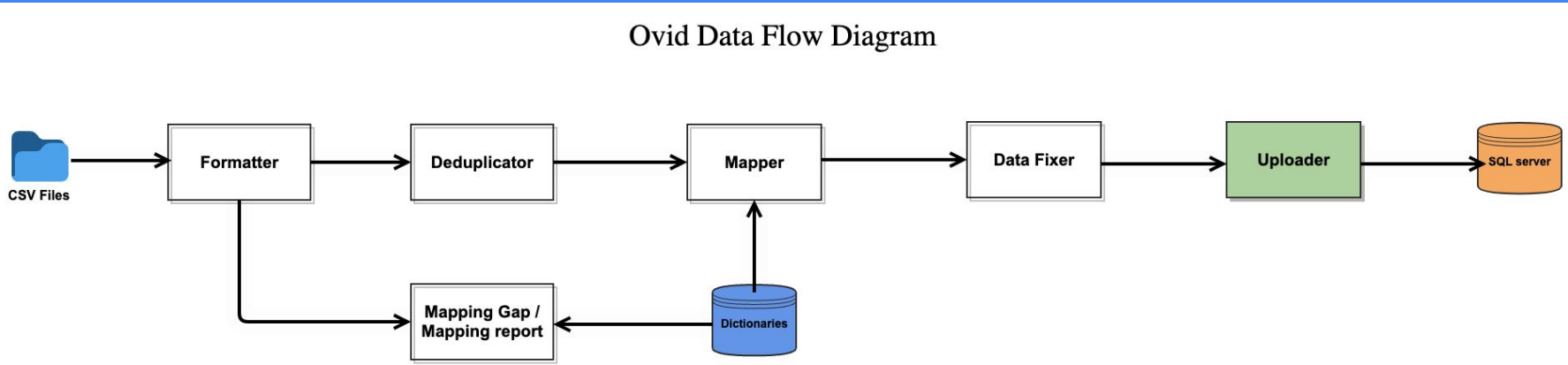
- Partner : -p [partner_name]
- Job : -j **fix**
- Folder : -f [folder_name1 folder_name2 …]
- Table : -t [table_name1 table_name2 …]

Example 1: Map the demographic and encounter tables:

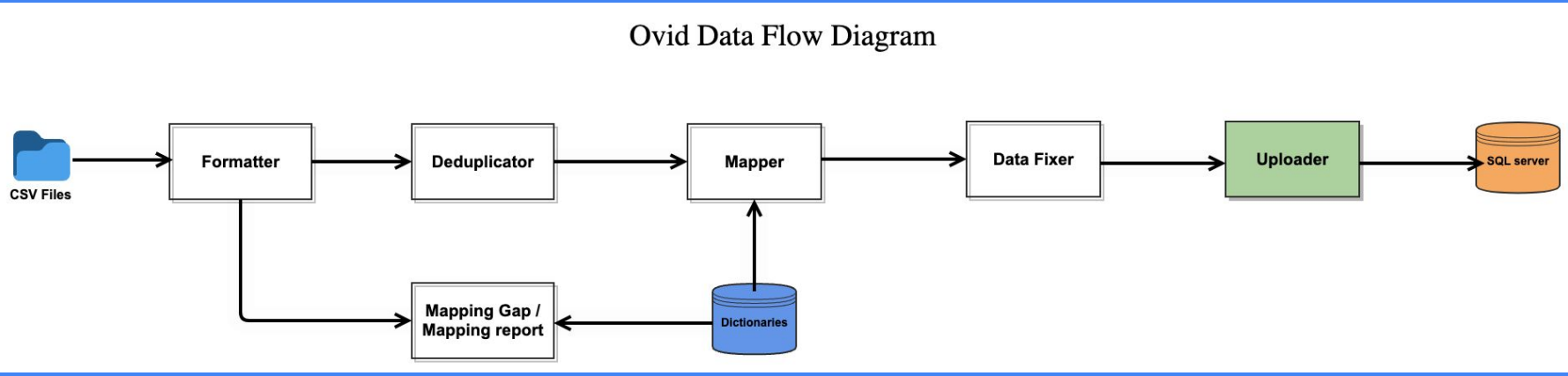
cluster run -a -- onefl_converter.py -p partner_a -j **fix** -t demographic encounter -f folder_1

Example 2: Map all the tables:

cluster run -a -- onefl_converter.py -p partner_a -j **fix** -t all -f folder_1



- Reads from the fixers output
- Ovid supports uploading data to the following databases types:
  - MSSQL
  - Postgres
  - Snowflake



Parameters:

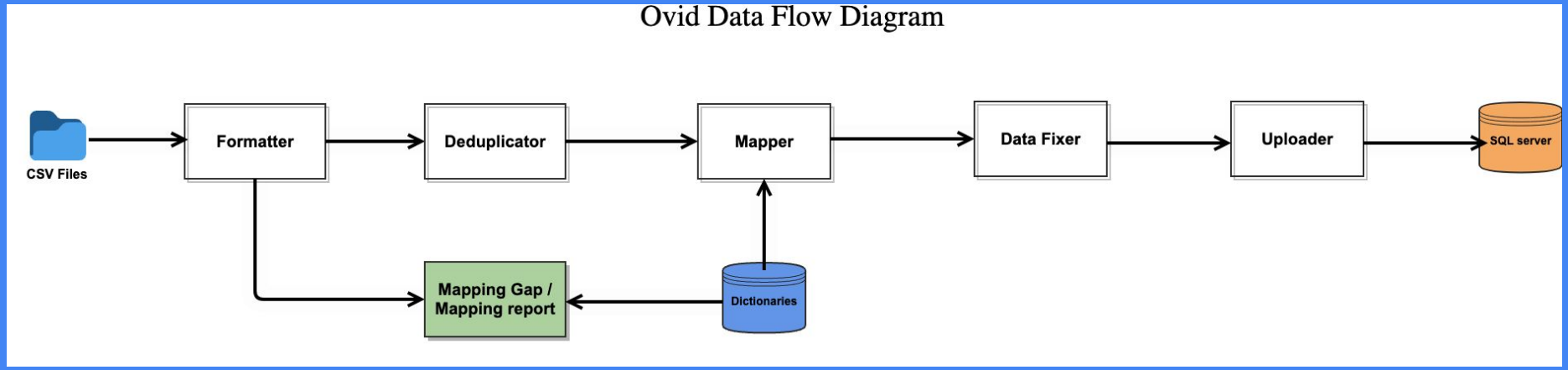
- Partner : -p [partner_name]
- Job : -j **upload**
- Folder : -f [folder_name1 folder_name2 …]
- Table : -t [table_name1 table_name2 …]
- Database name: -db [db_name]
- Database server: -s [server_name]
- Database type: -dt [sf, pg, or mssql]

Example 1: Deduplicate the demographic and encounter tables:

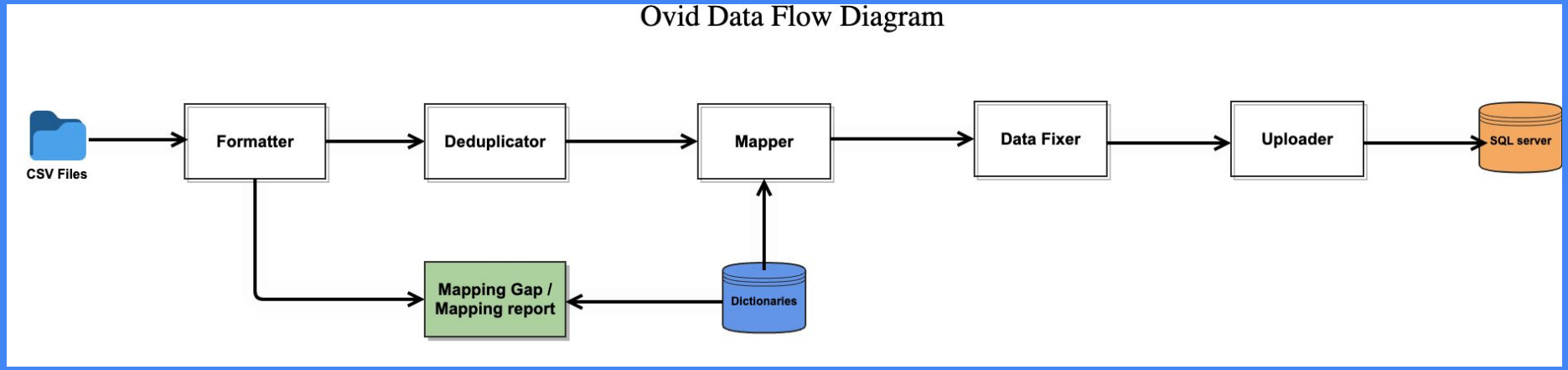
cluster run -a -- onefl_converter.py -p partner_a -j **upload** -t demographic encounter -f folder_1 -db [db_name] -s [server_name] -dt sf

Example 2: deduplicate all the tables:

cluster run -a -- onefl_converter.py -p partner_a -j **upload** -t all -f folder_1 -db [db_name] -s [server_name] -dt mssql



- Reads from the formatter output
- Compares the data to the existing dictionaries
- Create reports of any existing mapping gap between the input data and the dictionaries



Parameters:

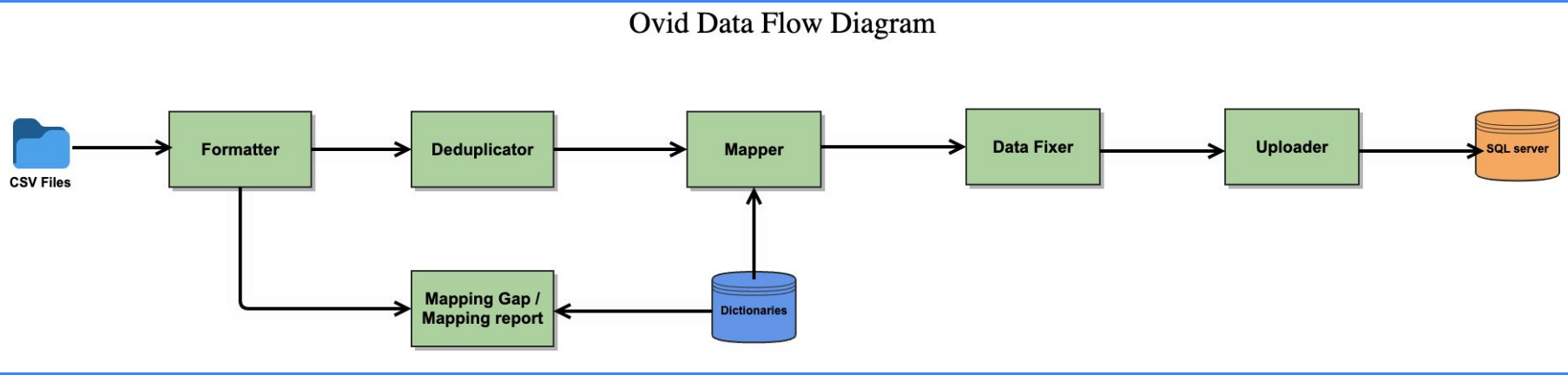
- Partner : -p [partner_name]
- Job : -j **mapping_gap**
- Folder : -f [folder_name1 folder_name2 …]
- Table : -t [table_name1 table_name2 …]

Example 1: Mapping gap for the demographic and encounter tables:

cluster run -a -- onefl_converter.py -p partner_a -j **mapping_gap** -t demographic encounter -f folder_1

Example 2: Mapping gap for all the tables:

cluster run -a -- onefl_converter.py -p partner_a -j **mappin_gap** -t all -f folder_1



Parameters:

- Input directory: -d [/path/to/data/parent/folder/]
- Partner : -p [partner_name]
- Job : -j **all**
- Folder : -f [folder_name1 folder_name2 …]
- Table : -t [table_name1 table_name2 …]
- db name : -db [db_name]
- Server name : -s [server_name]
- db type: : -dt [sf, pg, or mssql]

Example : run all the jobs at once:

cluster run -a -d [/path/to/data/parent/folder/] -- onefl_converter.py -p partner_a -j **all** -t all -f folder_1 -db [db_name] -s [server_name] -dt [sf, pg, or mssql]

Thank you!!

- Questions?

Github:

https://github.com/uf-hobi-informatics-lab/converter_2_0