1. GRAPE DATA IMPORT

> **Note!** This feature was introduced in Grape 0.0.9

The data import feature in Grape allows systems to import data from XLS or CSV format into the database. A pre-defined processing function can then be ran on all rows in the dataset, or the data can be "materialized" into a SQL table (test tables).

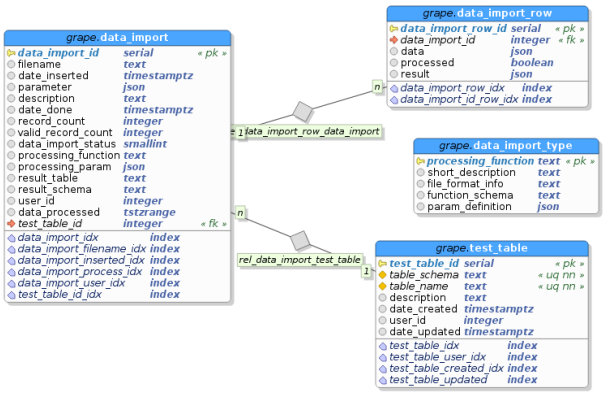

Fig. 1: Data Import Tables

The upload function will create separate tables for each file that is uploaded. The tables inherits from grape.data_import_row. It will be located in the schema set by data_import_schema, and the name stored in the column data_import.result_table. The rows, as it is imported in JSON format, will be stored as it is received.

Processing the file (running the pre-defined processing function) on the rows are done as a separate process. A call to ***POST /grape/data_import/process*** initiates this process. If dataimport_in_background is ***true***, the processing will happen in the background, through an internal background worker process (the name of this process is ***proc_process_data_import***).

1.1 Processing functions

Uploaded files can be registered against a pre-defined processing function. These functions must:

1. Return a JSON object, containing `{"result":{"status":"OK"}, "shared_data":{}}`. The utility functions ***grape.data_import_build_result*** can be used to build such an object
2. Accept two parameters: a grape.data_import object (containing all the information for the batch), and a JSON object (containing a ***data*** field with the row data, plus some additional information)
3. Be registered in grape.data_import using grape.upsert_data_import_type

```
CREATE OR REPLACE FUNCTION proc.dimport_generic (_data_import grape.data_import, _args JSONB)
    RETURNS JSON AS $$
DECLARE
BEGIN
    -- _data_import is a data_import record for the data_import_id that relates to this process,
    -- processing_param can be retrieved from this

        -- _args contains the following:
    --      data: the row data to be processed
    --      index: the index position of this process
    --      data_import_row_id: the data_import_row_id for this process
    --      shared_data: data accessable to all proccesses in their respective sequence
```

```
        -- The return data should be in the following format {"result":{"status":"OK"}}
        -- The result object is what will be stored as the result for processed row
        -- You can include shared_data if there is data you want to pass on to
    -- Proceeding processes

    -- {"result":{"status":"OK"}, "shared_data":{}}
        RETURN grape.data_import_build_result('OK');
END; $$ LANGUAGE plpgsql;
```

The function needs to be registered:

```
SELECT grape.upsert_data_import_type(
'dimport_generic', /* Function name */
    'Generic', /* Description */
    'This function does not actually process the data', /* File format information */
    'proc'); /* Function schema */
```

1.2 Test Tables

1.3 API calls

- POST /grape/data_import
- POST /grape/data_import/upload
- POST /grape/data_import/delete
- POST /grape/data_import/process
- GET /download/data_import/:data_import_id/:filename
- POST /grape/data_import/test_table/append
- POST /grape/data_import/test_table/delete
- POST /grape/data_import/test_table/alter

1.4 SQL functions

1.5 Grape settings:

- data_upload_schema
- dataimport_in_background