# **IGEM Tutorial**

# About the System

# Initial Settings

# Master Data Customization

# ETL process

# Monitor the workflow

# Data Maintenance

# Data Query

# Raise the process in the HPC environment

# About the System

The IGEM system was developed as a platform to host different APPs sharing common resources and interacting with each other through a single database.

The first APP created was GE (Gene x Exposome) to collect external data sets, identify words, and build a knowledge base available for dynamic and exploratory queries.

# Initial Settings

The IGEM system has a modular architecture divided into two groups: 1. Components that host services such as File System Management, Database, and WEB Interfaces; 2. APPs have been developed until now, the APP GE.

The GE APP hosts processes which are currently available:

* Data Workflow
  + collector
  + prepare
  + commute
  + mapreduce
* Master Data Load
  + loader
* Maintenance
  + maintenance
  + runserver
  + createsuperuser
  + makemigrations
  + migrate
* Filter and Queries
  + filter

We don't have an installation package yet; we need to copy the source files to a directory. IGEM is available in the ICDS environment at /gpfs/group/mah546/default/sw/igem, with the following folder structure:

* /igem\_db/: hosts the POSTGRES database structure and the system base in igemdb.
* /igem\_env/: hosts the conda virtual environment files.
* /src/: hosts the system and supports folders.

Inside the src directory, we will have:

* /ge/: all source codes and interfaces for the functioning of APP GE.
* /loader/: all input files for loading master data and output directory of the FILTER process.
* /psa/: Persist Store Area to store the database files downloaded and processed by the ETL process. Each DATASET will have its subfolder within the PSA.
* /src/: hosts the source code of IGEM components, configurations and parameterizations.
* /templates/: hosts the common web interfaces in IGEM.

The IGEM system is modular in terms of the database choice. The only problem is that it needs a setting, and it has simple user behavior. It is always necessary to instantiate it before its use. But in a cloud-as-a-service environment, it delivers better performance and multi-users compared to SQLite. For the ICDS project, the choice was POSTGRES, which has better characteristics for the need for GE.db, allowing better performance and security in HPC environments.

The database configuration is in the settings.py source file at :

Graphical user interface, text, application

Description automatically generated

# Master Data Customization

The master data have a primordial function for the system's functioning, being responsible for directing the origin of the data and for filtering and linking the words to the knowledge base.

Before starting the data collection, it will be necessary to parameterize the Master Data, which can be done via batch or individually in a web interface.

Master data will configure in the following order:

* + DATABASE
  + KEYGE-PREFIX
  + DATASET
  + KEYGE-GROUP
  + KEYGE-CATEGORY
  + KEYGE
  + KEYGE-WORD

Next, we will detail each master data.

**DATABASE**

A DATABASE record points to a specific external dataset, having authentication information, hosted data category, and description. It will reference in the DATASET (subsequent master data).

Access via the web will be at http://127.0.0.1:8000/admin/ge/database/

Batch data import will be through the command:

*$ Python manage.py loader --database.*

Before running the batch import command, the file must be saved in /loader/database.csv.

Batch import will disregard all repeated or existing records in GE.db.

**DATASET**

A DATASET record stores all information needed for external extraction and controls the links of related terms by the mapping process. The control of external data extraction occurs by DATASET. Each DATASET will consist of the fields:

* DATABASE: Grouping of DATASET, controlling access authentication.
* DATASET: Abbreviation for DATASET identification
* Description: Brief description of the purpose of DATASET
* Enabled: Flag informing if the DATASET is active and will be considered in the ETL process

Group Attributes: Fields to control extraction path and file type

* Source path from Internet: Flag to route the extraction path. If enabled, it will be via HTTP and disabled. It will be considered as a local file path.
* Source path: Path where the DATASET file is hosted
* Source file name: name of the file with the original data
* Source file format: file format with the original data. This information will be imported for conversion treatment to the data ingestion format in the ETL process. If compressed, inform only the compression format, type ZIP, GZ.
* Source file sep: inform the type of file separator if any. For tabular division, use /n
* Source file skip row: Inform the number of lines eliminated in the ETL process. Many files have structural information in their first lines that are not needed in the ETL process.
* Source compact: Flag to control if the file is compressed. If not marked, it is considered an uncompressed file.
* Target file name: Name of the file after unzipping
* Target file format: File format after unzipping, this field will be the actual file format, type CSV, TXT
* Keep file: Flag, if selected, will keep the file after data processing for future reference. It is essential to analyze the storage space consumption, as keeping the files may consume unnecessary space. New updates will overwrite existing files.

Group Columns: Controls rules for handling the extracted data in a format compatible with the GE.db system. Consider only columns with standard MEsH NIH codes.

* Column Sequence: Number of the column that receives the rule
* Column Name: Column name to guide and help identify the applied rule
* Active: a flag that informs if the practice is active
* PREFIX: inform the prefix of the word that will be considered and added to the column information.

Access via the web will be at http://127.0.0.1:8000/admin/ge/dataset/.

Batch data import will be through the command:

*$ Python manage.py loader –dataset*

In the batch import, the parameters of the transformation Group Columns will not be considered. After the batch import, accessing each DATASET via the web will be necessary, and parameterizing the column rules if they exist.

Before running the batch import command, the file must be saved to /loader/dataset.csv.

**GROUP**

The system groups will help in the type of search terms and will be assigned in the KEYGE explained later.

It has only two fields: Group and Description.

Access via the web will be at http://127.0.0.1:8000/admin/ge/group/.

Batch data import will be through the command:

*$ Python manage.py loader –group*

Before running the batch import command, the file must be saved to ../igem/src/loader/group.csv.

For our purpose, we will have two groups, environment and genomic.

**CATEGORY**

The categories help identify and search the terms and will be assigned in the KEYGE explained in the sequence.

It has only two fields: Category and Description.

Batch data import will be through the command:

*$ Python manage.py loader –category*

Before running the batch import command, the file must be saved to /loader/category.csv.

**KEYGE-PREFIX**

Groups to identify the type of information per column of extracted data. This information will be used when parameterizing the DATASET in the Columns group.

**KEYGE**

This master data will keep words that will be searched or converted in the ETL process of each DATASET.

The conversion will take place with the support of the KEYGE-WORD register, explained below.

The KEYGE will be the central point at the time of the searches, and the result of the ETL will be the connection of two KEYS.

KEYGE has the fields:

* KEYGE: search term
* Description: description of the term and aid in open and exploratory research
* Group: Grouping of terms to help with searches and grouping
* Category: Grouping terms at a second level to help with investigations and groupings.

Access via the web will be at http://127.0.0.1:8000/admin/ge/keyge/.

Batch data import will be through the command:

*$ Python manage.py loader –keyge*

Before running the batch import command, the file must be saved to ../igem/src/loader/keyge.csv.

In the file, inform the names of GROUP and CATEGORY. The process will identify the key automatically. If the values ​​of these two fields are not registered, an error message will be returned, and the load will not be performed. Proceed with the registration of the missing information before trying to load the KEYGE in batch.

**KEYWORD**

Master data that will keep the relationship of words and KEYGE. The system allows you to register compound words.

We will have the following fields:

* KEYGE: Sera is the term for conversion
* WORD: the word or set of words that convert to KEYGE
* ACTIVE: Flag to activate the relationship
* COMMUTE: Flag used to convert. If it is the same criterion between KEYGE and WORD, disable this flag to reduce memory consumption during the ETL process.

Access via the web will be at http://127.0.0.1:8000/admin/ge/keyword/.

Batch data import via command:

*$ Python manage.py loader -keyword*

# ETL process

The ETL process (Extraction, Transformation, and Loader) will be the step in which the system will fetch data from external sources, treat this data to a compatible standard, search for term relationships and write to GE.db. The entire process takes place based on briefly configured parameters and master data, so it is essential to pay attention to parameterization and master data entries.

For better resource management during the ETL phase, the process workflow was divided into five distinct phases:

**1st phase – Collect**

Process responsible for selecting the active DATASETS and for each one to check if a new version of the data is available. If so, it will extract the latest data and, if necessary, unzip the extracted file, leaving the file in a PSA structure (Persists Storage Area).

The process also performs the update of logs and version controls. The execution version of the steps in the web interface is still under development.

Execution of the process occurs in the command line, with script support to run all phases. Always run the commands inside the src folder of the program.

*$ python manage.py collect --run {all or dataset}*

It starts collecting the new datasets, downloading the latest data files from the internet or another location, and making them available in the PSA folder for the subsequent phases. If the call option is informed, the process will execute for all active DATASETs in the master data table or inform the specific DATASET for isolated execution.

*$ python manage.py collect --reset {all or dataset}*

The reset option will eliminate all phase control, including managing the last processed version, allowing reprocessing of the DATASET again. If you inform all choices, all active DATASETs will reset or notify only one DATASET for reset.

*$ python manage.py collect –show*

The show option will show all the registration DATASET and their status.

*$ python manage.py collect --activate {dataset}*

The activate option will activate a specific DATASET, allowing data extraction.

*$ python manage.py collect --deactivate {dataset}*

The deactivate option will deactivate a specific DATASET, preventing it from being part of future downloaded files.

**Phase 2 – Prepare**

This second phase of the process aims to transform the original data, thus reducing the need for computational resources in the subsequent steps. Based on the briefly configured DATASET parameters, in this phase, we will have:

* Deleting header lines
* Deleting unnecessary columns
* Transforming ID Columns with Suffix Identifiers
* Replacement the terms
* Deletion of the original file

The output will be a new temporary file for consumption in the next phase.

*# python manage.py prepare --run {all or dataset}*

It will start the data preparation phase for all DATASETs or just one specified. Essential to have the file in PSA. Otherwise, the system will display a warning.

*$ python manage.py prepare --reset {all or dataset}*

The reset option will reset the control for all or a specific DATASET in the preparation phase and the two later ones.

**Phase 3 – Commute**

In this phase, the process will replace words with terms (keyge) based on the previously parameterized Keyword record.

*# python manage.py commute --run {all or dataset}*

It will start the data term switching phase for all DATASETs or just one specified. Essential to have the file in PSA. Otherwise, the system will display a warning.

*$ python manage.py commute --reset {all or dataset}*

The reset option will reset the control for all or a specific DATASET to the switching phase and the next phase

**Phase 4 – MapReduce**

Last step of the process. It has a mechanism to find Keyges (terms) per line called Mapper and then activate the Reducer subprocess that will count the number of links found in the dataset. After all processing, the result will be recorded in the Keylinks table. It is important to note that the new data will fully replace the previous data in the processed dataset.

*$ python manage.py mapreduce --run {all or dataset}*

Essential to have the file in PSA. Otherwise, the system will display a warning. It will start the MapReduce phase of data terms for all DataSETs or just a specific one. In this phase, there is a large consumption of memory and processing, so it will be essential to allocate resources compatible with the size of the processed data.

*$ python manage.py mapreduce --reset {all or dataset}*

Reset option will Press the control to all or a specific DATASET in the current phase.

In all commands with run argument, possible multiprocessing, and control for file chunks. However, it will be necessary or necessary between the size of the extracted files and the resources allocated, such as memory and the amount of proposed balancing.

# Monitor the workflow

In addition to the reset commands shown above to control the process workflow, the system also has a web interface where the user can consult and manage the flow and status of the phases.

Browse http://127.0.0.1:8000/admin/ge/wfcontrol/ or select the Dataset – Workflow option in the GE application

In the first one, DATASETs that have already been started will be started, with the following references:

* DS STATUS: informs whether the DATASET is active or not for processing the 4 phases
* DATASET: Abbreviation for DATASET
* Last Update Dataset: Date of the previous data update.
* Source file version: Version of the final processed file.

The following four columns display the statuses still of the processes by phase. The green symbol indicates the status-completed successfully and the group not processed.

Data were only available in the GE.be database after all phases had been successfully executed.

A screenshot of a computer

Description automatically generated with medium confidence

When selecting a DATASET, it will be a details screen allowing the specific opening by field.

Graphical user interface, text, application

Description automatically generated

Important: it will not be necessary to include new DATASETs in the Workflow monitor. The system will automatically create a new control after the first load of data from the DATASET. If one of the workflow records is deleted, it will also be completed after the next data load.

# Data Maintenance

Help with the management tasks on the Database.

The first complete table deletion function is now available:

*$ python manage.py maintenance –delete {table}*

# Data Query

The GE.filter module is under development, but it is available for quick access. The filter command wants to extract the contents of a particular table and write it to an output file in the /loader/out\_from\_filter.csv directory.

*$ python manage.py filter –{table}*

Table available:

* Dataset
* Database
* Keyge
* Keylink
* Group
* Category
* Keyword

An interface is being developed to filter the KEYLINK table and command lines.

# Raise the process in the HPC environment

The IGEM system hosts the GE.db and GE.filter modules and is available in the ICDS environment for everyday use.

This document aims to demonstrate how to start IGEM and operate in command line and web interface environments.

**Start the system:**

Open an ICDS session and run the following commands:

*$ module load anaconda3*

*$ source activate /gpfs/group/mah546/default/sw/igem/igem\_env*

*$ cd /gpfs/group/mah546/default/sw/igem/igem\_db*

*$ pg\_ctl -D igemdb -l logfile start*

After these commands, we will have the session running a virtual environment with Anaconda3 running Python 3.10. We also activated the Database in Postgres. To check if everything is operational:

Check if you are running Python 3.10.4​​​​​​​

*$ Python -V*

Check if the database services are running

*$ ps aux | grep postgres*

IMPORTANT: once the Database services start, it will no longer be required to initialize again if you use the system in different ICDS sessions. However, due to the HPC architecture, it will only be possible for a user or job to activate the services. Therefore, after used the system, disable the database services as follows:

*$ ps aux | grep postgres*

*Graphical user interface, text

Description automatically generated*

*Disable the service that points to the base path (the case above: ID 5775)*

*$ kill {id}*

**Command Line**

The IGEM system has a master program that performs all operations triggering processes and parameters.

To run the program + process, we will go to the folder:

*$ cd /gpfs/group/mah546/default/sw/igem/src*

once in the folder we will type:

*$ python manage.py {process} (parameters}*

We have so far developed the process:

* loader
* collect
* prepare
* switch
* mapreduce
* createsuperuser
* runserver

Example of a complete command:

*$ python manage.py collect --show*

*$ python manage.py collect --run all*

*$ python manage.py collect --reset all*

The runserver process will be a session that will only run in an interactive RHEL7 session explained in the next topic

**Interface Web**

In the ICS-ACI Portal create a session in the RHEL7 interactive Desktop:

After the session is available, access it via Launch noCNV in the New tab button:

Graphical user interface, text, application

Description automatically generated

Go to the Mate Terminal:

A screenshot of a computer

Description automatically generated

Load the virtual environment:

*$ module load anaconda3*

*$ source activate /gpfs/group/mah546/default/sw/igem/igem\_env*

*$ cd /gpfs/group/mah546/default/sw/igem/src*

If the database has not started previously, you can check with the command:

*$ ps aux | grep postgres*

And start

*$ cd /gpfs/group/mah546/default/sw/igem/igem\_db*

*$ pg\_ctl -D igemdb -l logfile start*

With the conda virtual environment and database running, go to the src folder and run:

*$ python manage.py runserver*

Text

Description automatically generated

Go to the browser (firefox) and access <http://127.0.0.1:8000/admin>

            Username: igem / Password: igem

Graphical user interface, text, application, email

Description automatically generated

Within IGEM's administrative interface, we can navigate through the system's tables and perform operations such as adding new datasets, new keyge, mapping with words, or even consulting the processed links.

Graphical user interface

Description automatically generated

Important: Extracting, transforming, and loading the external data to the GE base through JOBs in the ICDS environment using multiprocessing.

 After used the system, disable the database services as follows:

*$ ps aux | grep postgres --color=auto postgres*

A screen shot of a computer

Description automatically generated with low confidence

*Disable the service that points to the base path (the case above: ID 5775)*

*$ kill {id}*

*$ conda deactivate*