Gen3 Data Modeling

Sheepdog & Peregrine

Herding Data Submissions & Hunting Down Data

Thursday, May 9, 2019
1:00 PM-2:00 PM (CST)
Gen3 Data Modeling

Herding Data Submissions & Hunting Down Data

Chris Meyer, Ph.D.

Center for Translational Data Science, University of Chicago

May 9, 2019
1. What is a Data Model?

1. Structure of a Gen3 Data Model

1. Herding Data Submissions: Data Import and Export

2. Hunting Down Data: Querying and Filtering Data

1. Demonstration of Query, Export, and Import in Workspace
What is a Data Model?
What is a Data Model?

- A **data model** organizes terms in a data dictionary and defines how they relate to one another. It is the implementation of a data dictionary and enables Gen3 services to **submit** and **query** data.
What is the Data Dictionary?

- The **data dictionary** defines and describes how research datasets are represented in the database and harmonizes term definitions from different data sources.

- **Data harmonization** is foundational to the *data commons* concept of sharing data for cross-project analyses.
What is the Data Dictionary?

- Dictionaries get everyone on the same page:
  - Define nodes and properties used across different but similar projects in a process called *data harmonization*.
  - Help avoid inconsistencies in data reporting and use across projects.
  - Make data easier to find, subset and analyze by enforcing Data Standards.
  - Support mapping terms to external controlled vocabularies like the NCIt, the National Cancer Institute’s Thesaurus.

**Example:** Different studies have their own unique term for “the date a participant enrolled in a clinical trial”. Those terms are *harmonized*, or mapped a single term, in the data dictionary.
The data model enables Gen3 services to **import**, **export**, and **query** data.

- **Data import and export** is accomplished by the *Sheepdog* service, which checks submissions against the data model to ensure all required fields are present and have appropriate values.

- **Database queries** are facilitated by the *Peregrine*, *Arranger*, and *Guppy* services. Queries must conform to the data model for successful data retrieval.
Structure of a Gen3 Data Model
The Gen3 Data Model is a graph-like relational model consisting of interrelated nodes that store certain related properties.
Structure of the Gen3 Data Model

- Structured Data are imported and exported as key-value pairs by Sheepdog.
- The data element keys are termed **properties** in Gen3.
- Property values can be queried using GraphQL, which is accomplished via the Peregrine, Arranger, or Guppy services.
- Sets of values in a node are called **records** or **entities**, which are assigned unique IDs (UUIDs).
Structure of the Gen3 Data Model

- Properties are organized into nodes, which are categories of structured data.
- Each node must have a relationship to at least one other node.
- The root node is program and must have the project node as its child.
Structure of the Gen3 Data Model

- The data model is a JSON created from node schemas in the YAML format.
- Each node is defined in a single schema.
- The schema contains the following:
  - A node **id** used for data query/submission.
  - A **category** used to group nodes conceptually.
  - A **description** which describes the node’s contents.
  - List of **links** defining relationship to other nodes.
  - List of **required** properties.
  - List of **properties**.
Structure of the Gen3 Data Model

● Property definitions include:
  ○ **property name** (e.g., “blood_tube_type”)
  ○ **description**
  ○ **type**
    - string
    - enum (enumerated values)
    - integer (whole numbers)
    - number (floats / numbers w decimal)
    - boolean (True/False)
    - array (a list of strings)
Structure of the Gen3 Data Model

- Limitations can be put on acceptable property values:
  - Minimum/maximum for integers/numbers.
  - Enumerations are limited strings.
  - Strings can be required to match patterns.
- Submitted records that do not conform fail.
Herding Data Submissions

*The Submission Service*
Herding Data Submissions

- The Sheepdog service shepherds submissions of structured data into the graph database.
- Sheepdog checks validity of each record in a data upload against the data dictionary to ensure all required fields are present and have appropriate data values.
- Sheepdog also supports export of structured data records in TSV or JSON formats.
Herding Data Submissions: Types of Data

- **Data files** must be downloaded to view its content, which is not accessible via API queries. Examples are images, tabulated data spreadsheets, or DNA sequencing reads.

- **Structured data** (AKA metadata) consists of records containing variable key-value pairs, which can be queried and modified via the API or viewed in Gen3 data exploration tools.
1. **User Authorization**

1. **Data File Upload:**
   a. Prepare Project in Submission Portal
   b. Upload Data Files to Object Storage
   c. Map Uploaded Files to a Data File Node

1. **Structured Data Submission:**
   a. Submit Structured Data
   b. Link Data File Records to their structured data
● The **Linked Data Lake** paradigm:
  ○ Data files are uploaded to object storage (AWS s3 bucket). Users don’t see bucket contents.
  ○ Indexd assigns a unique identifier called **GUID** to each file. Users access files via GUIDs.
  ○ Files in the **data lake** are **linked** to structured data using **GUIDs**
The `gen3-client` is a command-line tool for uploading and downloading data files:

- The client is configured with your credentials and sends files to an s3 bucket.
- A unique GUID is minted for each file.
- `Indexd` creates records linking the s3 locations of files with `data_file` records in the data model.
1. Configure the gen3-client with Credentials Downloaded from Windmill
   - `gen3-client configure --profile=profile_name --apiendpoint=https://nci-crdc-demo.datacommons.io/ --cred=~/.Downloads/credentials.json`

2. Upload files using the profile by passing the client a file location / RegEx
   - `gen3-client upload --profile=profile_name --upload-path=path/to/file.txt`
3. The final step in File Upload is **mapping the files to a node** in the model.

1. Click “Map my Files” in Windmill.
2. Choose files via checkbox to map to a particular node.
3. Assign values to required properties for the files.
4. Sheepdog creates the structured data records.
Structured Data Upload: Overview

- Now that files are uploaded and mapped to the data model, the rest of the project’s **structured data** must be submitted and linked to the data file records.
  - Sheepdog ensures **structured data** conform to the data model, and values can be queried via Peregrine
  - **Data files** on the other hand must be downloaded from object storage to view contents/values.

- Structured Data is submitted node-by-node.
- Typically data is submitted in TSV files (also accept JSON format).
- Sheepdog services checks submissions against the data model and creates one record for each row in a TSV (or entity in a JSON).
- Records are updated if a row has a previously created `submitter_id` or UUID
Structured Data Upload: TSV Submission Process

● TSV Submission Process

1. Download a template TSV for each desired node in your project.

2. Populate template TSVs with structured data.

3. Submit TSVs in the proper order (top-down, starting with the root node and moving towards “leaf” nodes).

4. Update links in data file TSV to link files to their corresponding, upstream structured data.

5. Sheepdog updates the records.
Structured Data: TSV Submission Troubleshooting

- During TSV submission, Sheepdog checks each entity (row in TSV) against the data dictionary.
  - TSVs are submitted in “chunks” of 30 records / rows
  - If any entity / row in the TSV is invalid with respect to the data model, the chunk will fail
After fixing the errors, the submission is successful and records are created or updated by Sheepdog.

- If an existing submitter_id or id is submitted, the record is updated instead of created.
- If data changes, the values are overwritten.
Hunting Down Data

Querying and Filtering Data
Windmill’s Exploration Page

a graphical user interface for cohort selection

Exploration
Windmill’s Exploration Page

- Cohorts can be selected via a graphical user interface using data facets.

- Once a cohort is selected, a file download manifest can be sent to your Workspace / JupyterHub for easy data file access and analysis.
API Queries for Cohort Building

Peregrine, Arranger, and Guppy
The GraphiQL interactive query building interface makes queries more intuitive for both Flat and Graph models

- Built-in documentation
- Autocomplete for objects, fields, arguments
- Ability to pass variables
Windmill’s Query Page

- Switch between the Flat and Graph models on Windmill’s “Query” Page.

- These use different endpoints that query different databases:
  - Graph Model hits the PostgreSQL DB
    vpodc.org/api/v0/submission/graphql/
  - Flat Model hits the ElasticSearch DB
    vpodc.org/api/v0/flat-search/search/graphql
Gen3 Query Overview

- **Graph Model**
  - **Peregrine** searches the **PostgreSQL** (graph database).
  - **Peregrine** translates GraphQL query to SQL.

- **Flat Model**
  - **Arranger/Guppy** searches the **ElasticSearch DB**.
  - Arranger translates GraphQL to ElasticSearch query.
  - ES queries support Aggregations.
  - **Guppy** facilitates easier GraphQL-like queries of ElasticSearch DB.
Flat Model: Aggregation Query

Flat Model queries support Aggregations for string and numeric fields:

- For strings:
  - bin counts - the number of records that have each key.

- For numeric fields:
  - summary statistics - minimum, maximum, average, count and sum.
The GraphQL Endpoints

- Queries can be sent to both flat and graph API endpoints programmatically.
Data Import and Access in the Gen3 Workspace

Import, Export and Query in the Gen3 Workspace JupyterHub
Data Import and Access in JupyterHub

- Data can be exported programmatically in, for example, a Python notebook using **the gen3-sdk**, which is an open-source suite of functions for interacting with Gen3 APIs.
- Import the gen3sdk in Python using “import gen3”
- The gen3sdk code lives on GitHub: [https://github.com/uc-cdis/gen3sdk-python](https://github.com/uc-cdis/gen3sdk-python)
● Now, we will take a look at the Gen3 Workspace, featuring data query, export, and import in JupyterHub
Future of Services
Guppy GraphQL: introduction

- *Simple* GraphQL schema to explore Flat model.
- But *powerful*, support everything Arranger does and more:
  - Histogram with bin aggregation for numbers;
  - No 10000 results limit;
  - JSON-based filters.

Future plans:
- Tiered access;
- Support searching by ontology values and it’s synonyms;
- SQL syntax for filters;
- Full-text search.

Not this:
```json
{
  subject {
    hits {
      total
g  
      edges {
        node {
          id
          race
        }n
      }
    }
  }
}
```
Export clinical data to PFB

Portable Format for Biomedical Data

Wait on Export to finish, it will export all filtered and available clinical data

Your PFB export is in progress. It may take up to 15 minutes to complete. Do not close your browser until your PFB export is finished.

After some time it will provide a copyable URL to PFB export of the clinical data

Your cohort has been exported to PFB! The URL is displayed below.
Most recent PFB URL: https://ctds-testing-url.s3.amazonaws.com/some-folder/pfb.avro
Learn More

- [github.com/uc-cdis](https://github.com/uc-cdis)
- [gen3.org](https://gen3.org)
- Gen3 Community on Slack
- [support@datacommons.io](mailto:support@datacommons.io)
- [ctds.uchicago.edu](https://ctds.uchicago.edu)
Selected Data Commons Using Gen3

- NIH National Cancer Institute
  Cancer Research Data Commons
- Kids First Pediatric Research Program
- Data Stage
- NIAID Data Hub Pilot
- National Heart, Lung, and Blood Institute
- Brain Commons
- National Human Genome Research Institute
- AnVIL
Next Webinar

Data Science with Gen3*
Using Jupyter Notebooks

Thursday, June 13, 2019  1:00 PM - 2:00 PM (CST)
Questions?