GenQuery2: A more standardized, powerful parser for the iRODS namespace

Kory Draughn
Chief Technologist
iRODS Consortium

June 13-16, 2023
iRODS User Group Meeting 2023
Chapel Hill, NC

Overview

- What is GenQuery2?
- GitHub Repository
- General Features
- Components and Examples
  - API Plugin
  - Rule Engine Plugin
  - iCommand
- Remaining Work
- Future Plans
- Community Engagement

What is GenQuery2?

An experimental redesign (and implementation) of the iRODS GenQuery parser.

This project exists as a means for allowing the iRODS community to test the implementation and provide feedback so that the iRODS Consortium can produce a GenQuery parser that is easy to understand, maintain, and enhance all while providing a syntax that mirrors standard SQL as much as possible.

Once stable, the code will be merged into the iRODS server making it available with future releases of iRODS.

https://github.com/irods/irods_api_plugin_genquery2

The repository contains all source code for generating a package containing the following ...

- An API Plugin
- A Rule Engine Plugin
- An iCommand

Everything discussed in this talk can be found in the repository.

General Features

- Enforces the iRODS permission model
- Logical AND, OR, and NOT
- Grouping via parentheses
- SQL CAST
- SQL GROUP BY
- SQL aggregate functions (e.g. count, sum, avg, etc)
- Per-column sorting via ORDER BY [ASC|DESC]
- SQL FETCH FIRST *N* ROWS ONLY (LIMIT offered as an alias)
- Metadata queries involving different iRODS entities (i.e. data objects, collections, users, and resources)
- Operators: =, !=, <, <=, >, >=, LIKE, BETWEEN, IS [NOT] NULL
- SQL keywords are case-insensitive
- Federation is supported

Components and Examples - API Plugin

Wraps the parser and makes it available to all clients.

Interface Details

- API Number
  - 1000001 *(may change in the future)*
- Input Parameters
  - query_string - The GenQuery2 string.
  - zone - The name of the zone to execute the query in.
  - sql_only - An integer instructing the plugin to return the generated SQL.
- Output
  - On success, returns a JSON string representing the resultset
  - On failure, returns an iRODS error code

Defaults to returning a max of 16 rows if the client does not specify the number of rows to return.

Makes GenQuery2 available to the iRODS Rule Language and other rule engine plugins.

The use of a rule engine plugin is temporary, but required for 4.3.0. This requirement will be lifted following the release of iRODS 4.3.1.

The rule engine plugin includes the following rules ...

- genquery2_execute(**handle*, **query_string*)
- genquery2_next_row(**handle*)
- genquery2_column(**handle*, **index*, **value*)
- genquery2_destroy(**handle*)

Components and Examples - Rule Engine Plugins Example

Enable access to the rules by adding the following to the rule_engines stanza of server_config.json. For example ...

```
{
    "instance_name": "irods_rule_engine-genquery2-instance",
    "plugin_name": "irods_rule_engine-genquery2",
    "plugin_specific_configuration": {}
}
```

Example rule ...

```
genquery2_test_rule()
{
    # Execute a query. The results are stored in the Rule Engine Plugin.
    genquery2_execute(*handle, "select COLL_NAME, DATA_NAME order by DATA_NAME desc limit 1");

    # Iterate over the resutls.
    while (errorcode(genquery2_next_row(*handle)) == 0) {
        genquery2_column(*handle, '0', *coll_name); # Copy the COLL_NAME into *coll_name.
        genquery2_column(*handle, '1', *data_name); # Copy the DATA_NAME into *data_name.
        writeLine("stdout", "logical path => [*coll_name/*data_name]");
    }

    # Free any resources used. This is handled for you when the agent is shut down as well.
    genquery2_destroy(*handle);
}
```

iquery - A binary which enables execution of GenQuery2 queries via the command line.

```
iquery - Query the iRODS Catalog

Usage: iquery [OPTION]... QUERY_STRING

Queries the iRODS Catalog using GenQuery2.

QUERY_STRING is expected to be a string matching the GenQuery2 syntax. Failing
to meet this requirement will result in an error.

Mandatory arguments to long options are mandatory for short options too.

Options:
      --sql-only         Print the SQL generated by the parser. The generated
                         SQL will not be executed.
  -z, --zone=ZONE_NAME   The name of the zone to run the query against. Defaults
                         to the local zone.
  -h, --help             Display this help message and exit.

iRODS Version 4.3.0                  iquery (experimental)
```

List the number of replicas for all data objects. **jq** is used for formatting purposes.

```
$ iquery "select COLL_NAME, DATA_NAME, count(DATA_ID) group by COLL_NAME, DATA_NAME" | jq
```

Below is the output from running the command.

```
[
  [
    "/tempZone/home/rods",
    "foo",
    "3"
  ],
  [
    "/tempZone/home/rods",
    "bar",
    "1"
  ]
]
```

Show the SQL generated by the parser. **pg_format** is used for formatting purposes.

```
$ iquery --sql-only \
      "select COLL_NAME, DATA_NAME, count(DATA_ID) group by COLL_NAME, DATA_NAME" | \
      pg_format -
```

Below is the output from running the command. **The SQL is never executed.**

```
SELECT DISTINCT
    t0.coll_name,
    t1.data_name,
    count(t1.data_id)
FROM
    R_COLL_MAIN t0
    INNER JOIN R_DATA_MAIN t1 ON t0.coll_id = t1.coll_id
    INNER JOIN R_OBJT_ACCESS pdoa ON t1.data_id = pdoa.object_id
    INNER JOIN R_TOKN_MAIN pdt ON pdoa.access_type_id = pdt.token_id
    INNER JOIN R_USER_MAIN pdu ON pdoa.user_id = pdu.user_id
    INNER JOIN R_OBJT_ACCESS pcoa ON t0.coll_id = pcoa.object_id
    INNER JOIN R_TOKN_MAIN pct ON pcoa.access_type_id = pct.token_id
    INNER JOIN R_USER_MAIN pcu ON pcoa.user_id = pcu.user_id
WHERE
    pdu.user_name = ?
    AND pcu.user_name = ?
    AND pdoa.access_type_id >= 1050
    AND pcoa.access_type_id >= 1050
GROUP BY
    t0.coll_name,
    t1.data_name FETCH FIRST 16 ROWS ONLY
```

Remaining Work

The following items must be resolved before making GenQuery2 a part of the server.

- Clean up the CMakeLists.txt file
- Implement tests
- Discuss how much GenQuery2 should know about Groups
  - https://github.com/irods/irods_api_plugin_genquery2/issues/3
- Discuss how much GenQuery2 should know about Tickets
  - https://github.com/irods/irods_api_plugin_genquery2/issues/4

Future Plans

- Expose more SQL features
  - CASE, HAVING clauses
  - Sub-selects
  - Multi-argument functions
- Consider controlling various options through GenQuery2 syntax
  - e.g. iquery "option distinct off; select DATA_NAME"
- Consider switching from boost::variant to std::variant
- Simplify pagination
  - Provide a utility library that manages the page information
  - Provide a document explaining how the utility may be implemented

Community Engagement

We are considering the idea of releasing GenQuery2 as an experimental package.

- Allows the community to try GenQuery2 and provide feedback
- Allows frequent updates (no ties to a server release)
- Does not target a specific version of iRODS

We need your feedback!

The more the community participates, the better GenQuery2 will become.

Thank you!

Questions?

If you're interested in learning more about the implementation and/or seeing more examples of GenQuery2, please watch this TRiRODS talk.