iCAT Interactions in iRODS

Wayne Schroeder
DICE/INC
University of California
San Diego, CA, USA.

Arcot (Raja) Rajasekar
DICE Center/SILS/RENCI
University of North Carolina
Chapel Hill, NC, USA.
Introduction

- Role of iCAT
- Internal Interactions: iCAT & iRODS
- User Interactions: iquest
- Rule Interactions:
  - micro-services
  - irule command
- Extensible ICAT
User Can Search, Access, Add and Manage Data & Metadata

Overview of iRODS Data System

*Access data with Web-based Browser or iRODS GUI or Command Line clients.
iRODS Distributed Data Management
input With Replication

Client

Resource 1

Resource 2

input

data

metadata

metadata

metadata
Role of iCAT in iRODS

- iCAT stores persistent information about all aspects of iRODS
- State Information about the whole data grid
- Provides a Mapping from Logical Names to Physical Names
  - Example: demoResc maps to a
    - Host-address: brick14.sdsc.edu
    - Vault Path in File: /data/g1/
- Stores usability Information such as ACLs, Audit trails, Quotas, Groups, ...
- iCAT is transactional -- all or none
Software Layers in iCAT

- **Low-level Functions** (Database-specific)
- **Mid-level Functions** (Glue: Major Workhorse procedures)
- **High-level Functions** (iRODS Interface Functions)
- **Micro-Services** (Policy Functions)
iCAT and iRODS

- Only One iCAT per iRODS Grid
- Information stored in vendor database: Postgres, Oracle, MySQL.
- Has a Very Rich Schema
  - User is transparent to this schema
  - User sees one humongous table
- Has an extensibility component
  - Helps in customizing iCAT to user needs
  - Can add new tables to iRODS Schema and use the same software framework
iRODS interacting with iCAT

- Specific Interactions
  - Special Function Calls Particular to the needs of iRODS
  - Example:

- General Interactions: Generic Calls based on single-table schema.
  - Query
  - Update: Insert, Delete, Modify
General Interactions

High Level APIs

- rsGenQuery
- rsGeneralRowInsert
- rsGeneralRowPurge
- rsGeneralUpdate
- rsGeneralAdmin  -- administrative api
- rsUserAdmin  -- special case for user
- rsModAVUMetadata  -- triplet api

Needed by core & micro-service developers
Easy way of Querying iCAT: iQuest (1)

- iCommand utility for querying the iCAT
- It is in pseudo-SQL format
  - SQL is a query language for databases
  - Stands for structured query language
- You view the whole iCAT as one large table
  (iCAT has more than 20 tables)
  - You give conditions for picking rows from the "universal" table
  - You give a list of column names to pick values in the rows
  - SELECT DATA_NAME
    WHERE DATA_NAME like '%.txt'
    AND COLL_NAME = '/myzone/home/me'
iQuest Column Names

- Found in rodsGenQueryNames.h
- Example:

  USER_NAME       USER_ZONE       USER_TYPE
  ZONE_NAME       USER_ID         RESC_ID
  RESC_NAME       RESC_LOC        RESC_VAULT_PATH
  RESC_STATUS     DATA_ID          DATA_NAME
  DATA_TYPE       DATA_PATH        DATA_RESC_NAME
  COLL_NAME       DATA_CHECKSUM
  DATA_COMMENTS   DATA_CREATE_TIME
  COLL_OWNER_NAME  COLL_ACCESS_NAME
  META_DATA_ATTR_NAME META_DATA_ATTR_VALUE
Easy way of Querying iCAT: iQuest (2)

- The iquest command:
  iquest [format] selectQuery

- Samples:
  iquest "SELECT DATA_NAME WHERE DATA_NAME like '%.txt'"
  iquest "File %s has %-2.2s copies"
  "SELECT DATA_NAME, DATA_REPL_NUM"

- Complicated Example:
  iquest "User %-9.9s uses %14.14s bytes in %8.8s files in '%s'"
  "SELECT USER_NAME, sum(DATA_SIZE),
   count(DATA_NAME), RESC_NAME"

User sekar has 25342 bytes in 342 files in demoResc
User sekar has 34529 bytes in 412 files in tapeResc
How to query in iRule (1)

- Two Micro-services:
  - `msiMakeQuery(*colList, *cond, *queryStr)`
    - Takes a list of columns and a condition string and creates a pseudo-SQL query-string
    - Alas! Does not do formats; but don’t despair!!
  - `msiExecStrCondQuery(*queryStr, *genQOut)`
    - Takes the query-string executes it in iCAT and returns the answer-table in an internal structure

- Sample-rule: Given a condition get the answer-table
  ```
  acExecMyQuery(*C,*T)||
  msiMakeQuery("DATA_NAME,COLL_NAME",*C,*S)##
  msiExecStrCondQuery(*S,*T) | nop
  ```

- But *T is an internal structure and not printable!!
How to query in iRule (2)

- So, to print,
  - we need to take the values out of the structure
    - `msiGetValByKey(*Row, *ColName, *Value)`
    - Given a row of the table, and a column name, it returns the value of that column.
  - How do we print a value?
    - `writeLine(*where, *what)`
    - `writeLine (stdout, "Hello World!")`
  - How to get a row from the table (of rows)
    - Use the `forEachExec` system micro-service

```
forEachExec(*T, msiGetValByKey(*T, DATA_NAME, *Value))##
    writeLine(File Name is *Value), nop)
```
Finally, we can put all together:

```plaintext
myRule(*Cond)
{
    msiMakeQuery("DATA_NAME,COLL_NAME",*Cond,*S);
    msiExecStrCondQuery(*S,*T);
    forEachExec(*T) /* for each row in answer table T */
    {
        msiGetValByKey(*T, DATA_NAME, *DV);
        msiGetValByKey(*T, COLL_NAME, *CV):
        writeLine(File *DV is in Collection *CV)
    }
}
```
Other interesting micro-services

- msiAssociateKeyValuePairsToObj
- msiExecStrCondQueryWithOptions
- msiGetContInxFromGenQueryOut
- msiGetMoreRows
- msiAddSelectFieldToGenQuery
- msiAddConditionToGenQuery
- msiPrintGenQueryOutToBuffer
- msiPrintGenQueryInp
- msiRemoveKeyValuePairsFromObj
- msiAssociateKeyValuePairsToObj
Extensible ICAT

Student and Department Tables

StuTab

<table>
<thead>
<tr>
<th>SName</th>
<th>DId</th>
</tr>
</thead>
</table>

DeptTab

<table>
<thead>
<tr>
<th>DId</th>
<th>DName</th>
</tr>
</thead>
</table>

HIDE BUT LINK INTERNALLY

External Names

STUDENT_NAME

STUDENT_DEPT
Extensible iCAT

- Modules/extendedICAT
- New Tables and their relationships are coded in extendedICAT.h

- Define Internal COLUMNS
  ```
  #define COL_ONE  100001
  #define COL_TWO  100002
  #define COL_ID1   100003
  #define COL_ID2   100004
  ```

- Define External Names for the COLUMNS
  ```
  { COL_ONE, "STUDENT_NAME"},
  { COL_TWO, "STUDENT_DEPT"},
  ```
Extensible ICAT

- Map Internal COLUMN to DB table
  - \{COL\_ONE, "stuTab", "SName"\},
  - \{COL\_TWO, "deptTab", "Dname"\},
  - \{COL\_ID1, "stuTab", "DId"\},
  - \{COL\_ID2, "deptTab", "DId"\},

- Map Links between tables for automatic SQL generation
  - \{"stuTab", "DId", "deptTab", "DId"\},
Querying Ext ICAT

- Query:
  iquest "SELECT STUDENT_NAME WHERE STUDENT_DEPT = 'sils' AND STUDENT_NAME like 'Terrel%'"

- Similar to:
  iquest "SELECT DATA_NAME WHERE COLL_NAME like '/home/sekar/%'"
One Department Per Student

<table>
<thead>
<tr>
<th>StuTab</th>
<th>DeptTab</th>
</tr>
</thead>
<tbody>
<tr>
<td>SName</td>
<td>DId</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>DId</td>
<td>DName</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Student In more than one Department

Hides schema changes (to some extant)
What Does Ext ICAT Buy Us?

- Can integrate with Core ICAT Tables.
  - Example:
    - Student’s can be iRODS Users.
    - Link STUDENT_NAME to USER_NAME
- Can link Data Objects with Standard Metadata Schemas cast in RDB
  - Example:
    - FITS metadata for astronomy images
    - DICOM for MRI
- Can Now Query on these Metadata Schemas and get associated data
- Extraction/Metadata Ingestion possible with micro-services
- No need to maintain another DB
Conclusion

- Role of iCAT
- Internal Interactions: iCAT & iRODS
- User Interactions: iquest
- Rule Interactions:
  - micro-services
  - irule command
- Extensible ICAT