university of groningen / center for information technology

Research Data Management at RUG

Centre for Information Technology

The Research Data Management System at the University of Groningen (RUG RDMS): architecture, solution engines and challenges

A. Tsyganov, S. Stoica, M. Babai, V. Soancatl-Aguilar, J. Mc Farland, G. Strikwerda, M. Klein, V. Boxelaar, A. Pothaar, C. Marocico, J. van den Buijs

(rdms-support@rug.nl)

university of groningen / center for information technology

RUG RDMS

- Facts and figures
  - Key figures
  - Position in international rankings
  - Annual Reports University of Groningen
  - Institutional budget
  - Allocation of funds

Key figures

- Founded in 1614
- 11 faculties (1 in the Frisian capital of Leeuwarden)
- 6,250 fte staff (including University Medical Center Groningen, UMCG)
- More than 120 nationalities
- 140,000 alumni
- 1,000 million EUR budget

Education (2020)

- 34,000 students
- 8,250 international students (24%)
- > 120 English-taught Master's degree programmes
- > 45 Bachelor's degree programmes, of which >35 taught in English

Research (2020)

- 3,600 fte academic staff (39% international)
- 425 full professors
- 4,350 PhD candidates (51% international)
- 591 PhD theses
- 8,000 research publications (dissertations not included)
- 18 patent applications

Excellent prizes and grants (2015-2020)

- Nobel Prize for Ben Feringa

university of groningen / center for information technology

https://www.rug.nl/about-ug/profile/facts-and-figures/

RUG RDMS: Agenda

- Storage architecture
- Application design
- Web Interface
- iRODS rules
- Custom policies engine
- Metadata templates
- Auditing
- What's next?
- Q&A

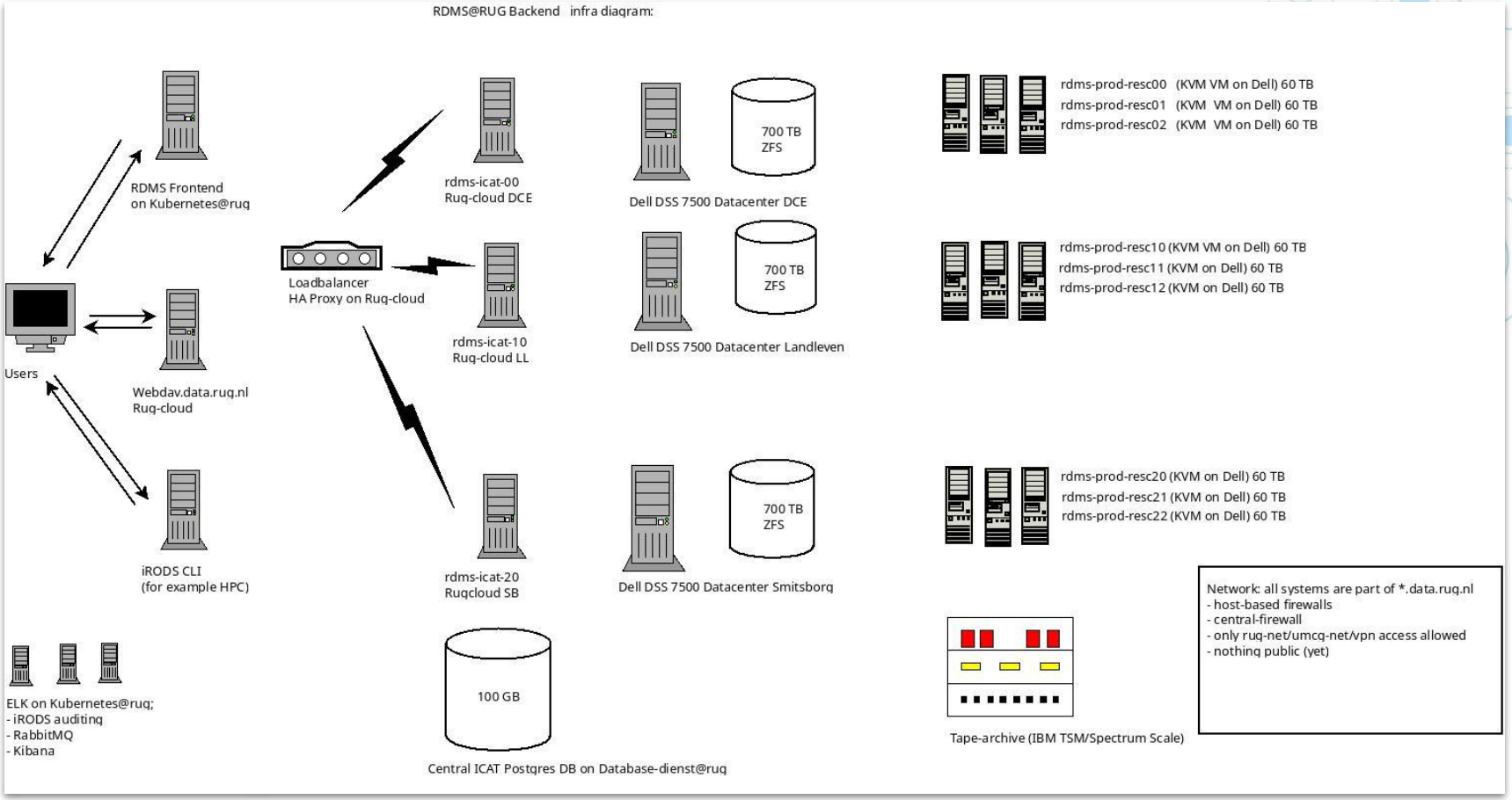
university of groningen / center for information technology

Storage Architecture

RUG RDMS: Storage Architecture

- our own "Google Drive" for Research-data (keep data on-premises)
- object store instead of old-school filesystems
- self service: user can set ACL's on files without helpdesk
- self service: user can update/add metadata to files
- home-data, group-data, project-data
- files/access can be audited (rabbitmq/kibana/ELK-stack)
- files are replicated (2 copies, spread between 3 locations)

RUG RDMS: Storage Architecture



RUG RDMS: Storage Architecture

3 'Big' storage-servers (located in 3 different DC's)

Ubuntu/ZFS: fancy/hip filesystem (data-deduplication, compression,
'self'-healing, software RAID

```
root@gargantua0:~# zpool status | grep sd | wc -l
90

root@gargantua0:~# zpool list
NAME   SIZE  ALLOC   FREE  EXPANDSZ   FRAG    CAP  DEDUP  HEALTH
pool0   616T  97.1T   519T                -     0%    15%  1.00x    ONLINE
```

RUG RDMS: Storage Architecture

IRODS Zones:

- rugZone: big zone for current projects
- umcgZone: dedicated zone for UMCG (storage on datahandeling)
- sramZone: dedicated (test) zone for SRAM (external users test)
- testZone: dedicated test zone for user-test/workshops
- devZone: dedicated dev zone for developers/playground

More/extra Zones are VM's,so they can run on the same central Infrastructure.

RDMS backend should survive..

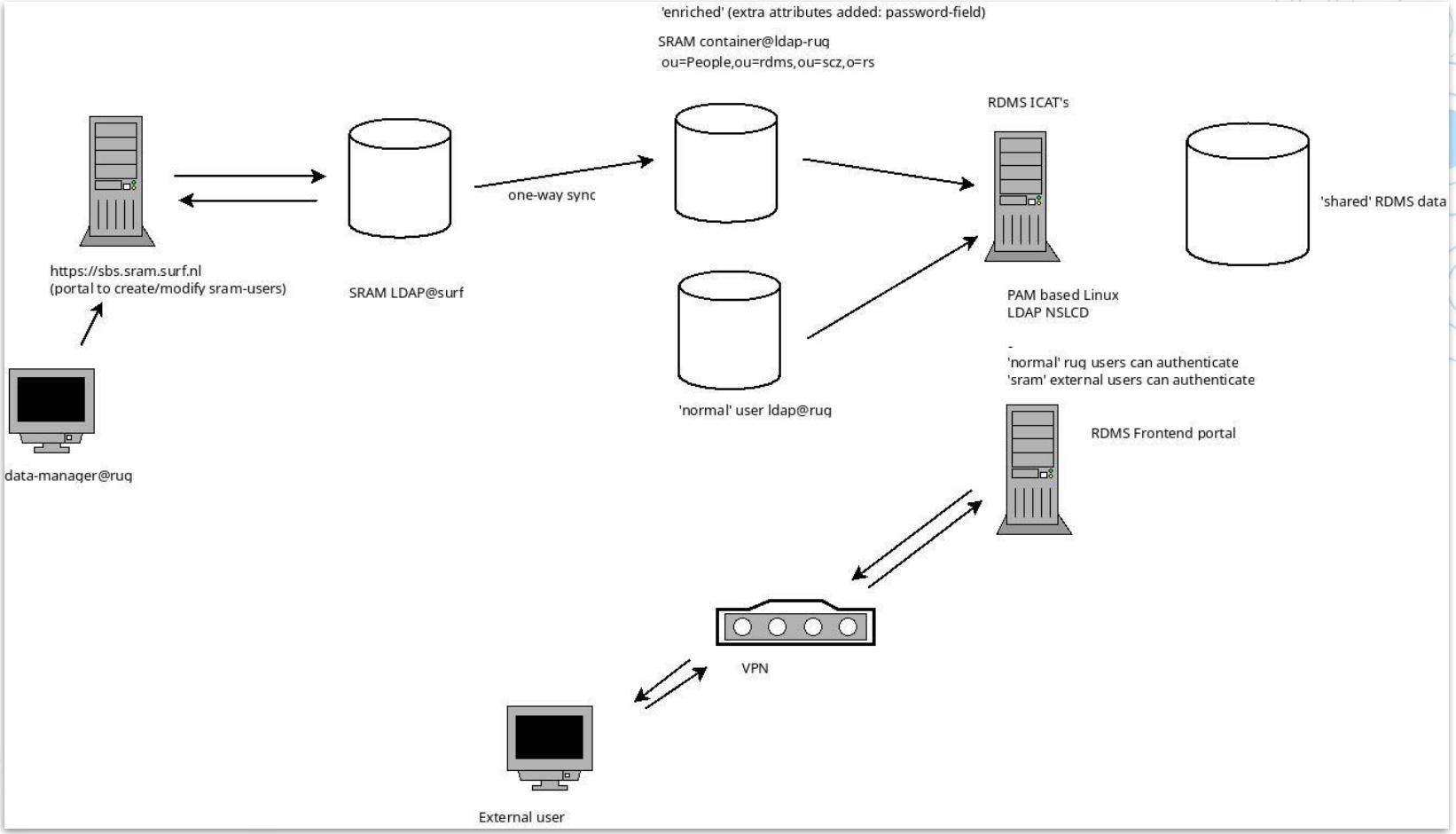
- broken disks (hot-swappable)
- broken volume (files are replicated)
- broken server(s)
- 1 downed data-centre
- small scale network outage
- maintenance...
- resc-vm's are small (move/migrated if needed)
- scalable workload (load-balancers, multi-site)
- High Performance Computing workloads

RUG RDMS: Storage Architecture

Authentication:

- PAM-based authentication stack
- login:email/university-password
- RUG-LDAP (rug-users)
- SRAM-LDAP (for external users)

RUG RDMS: Storage Architecture



RUG RDMS: Storage Architecture

Security best-practises:

- no root ssh login on systems
- SSL everything
- only 2 admins (admin-group) can ‘sudo’
- ssh-login only from admin-network (BWP-lan)
- host-based firewall (iptables)
- network based central firewall (Palo Alto)
- stealth (intruder detection/file integrity)
- fail2ban on webdav
- todo/nice-to-have: fail2ban on iRODS
- monthly security-audit on systems by sec team
- external company pentesting (yearly)

Application design

RUG RDMS: Application design

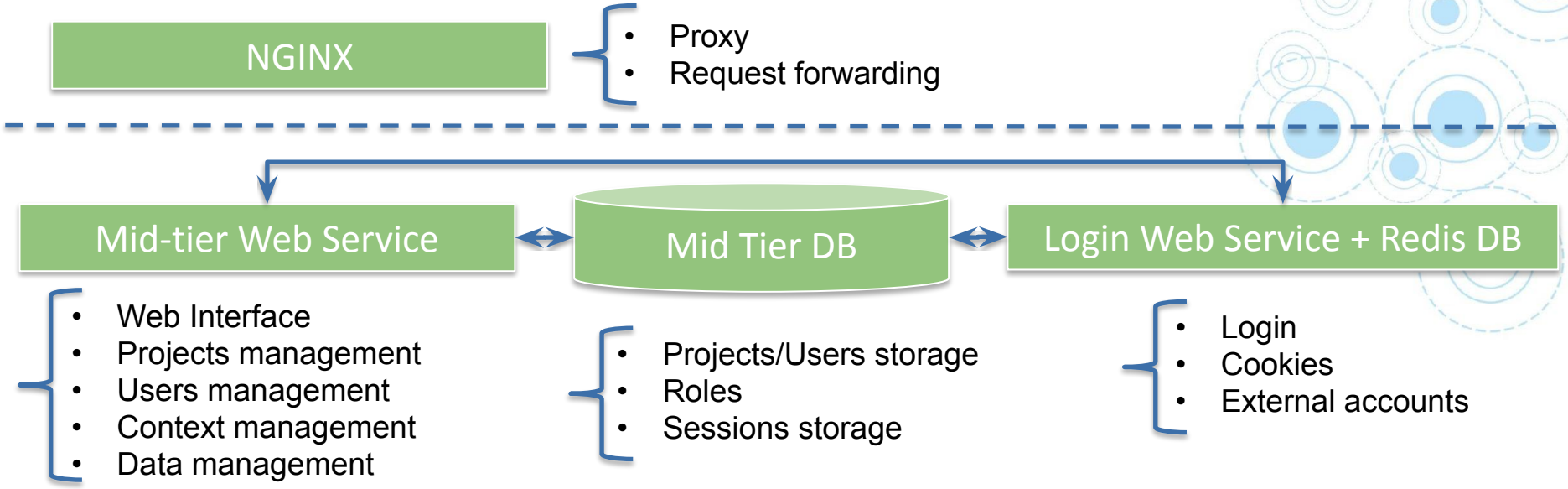
- Modularity
  - Different applications can make use of the system through an API
- Abstraction
  - Functionality is split hierarchically among the different layers: iRODS, mid-tier, front-end;
  - iRODS layer implements basic functionality and exposes it to higher layers through an API
- Containerization: Docker for services, VMs for iRODS/resource servers
- Robust: vertically and horizontally

RUG RDMS: Application design

NGINX

- Proxy
- Request forwarding

RUG RDMS: Application design



RUG RDMS: Application design

NGINX

- Proxy
- Request forwarding

Mid-tier Web Service

- Web Interface
- Projects management
- Users management
- Context management
- Data management

Mid Tier DB

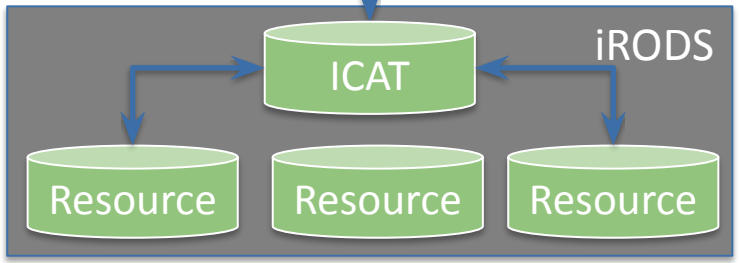
- Projects/Users storage
- Roles
- Sessions storage

Login Web Service + Redis DB

- Login
- Cookies
- External accounts

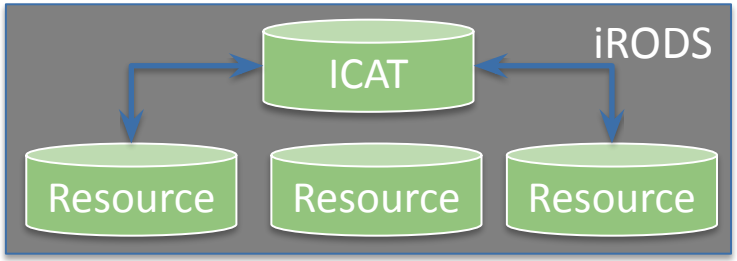
iRODS Web Service

- iRODS connection
- iRODS requests
- iRODS default search



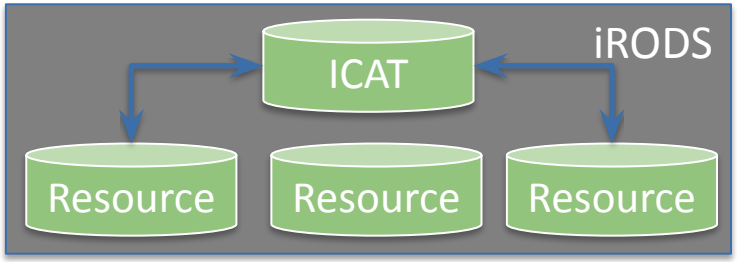
university of groningen / center for information technology

RUG RDMS: Application design



university of groningen / center for information technology

RUG RDMS: Application design



- command line interface
- Web Dav
- other...

university of groningen / center for information technology

RUG RDMS: Application design

Metadata

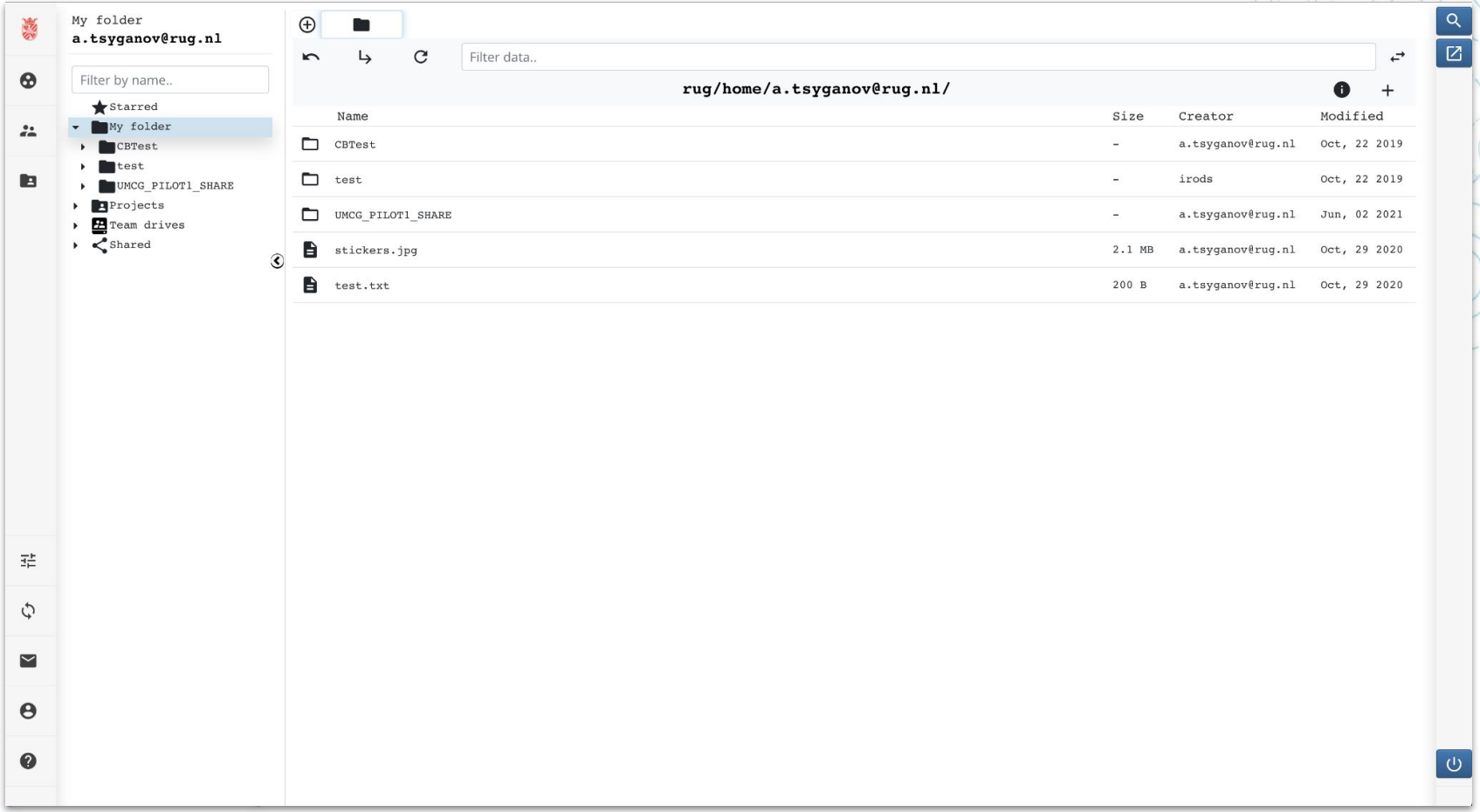
- Mutable metadata
  - User generated metadata from Web interface
  - User generated metadata from command line interface
- Immutable metadata
  - Automatic extracted metadata
  - System metadata

System metadata is a number of metadata attributes in different formats that are used by the RDMS to control data workflows

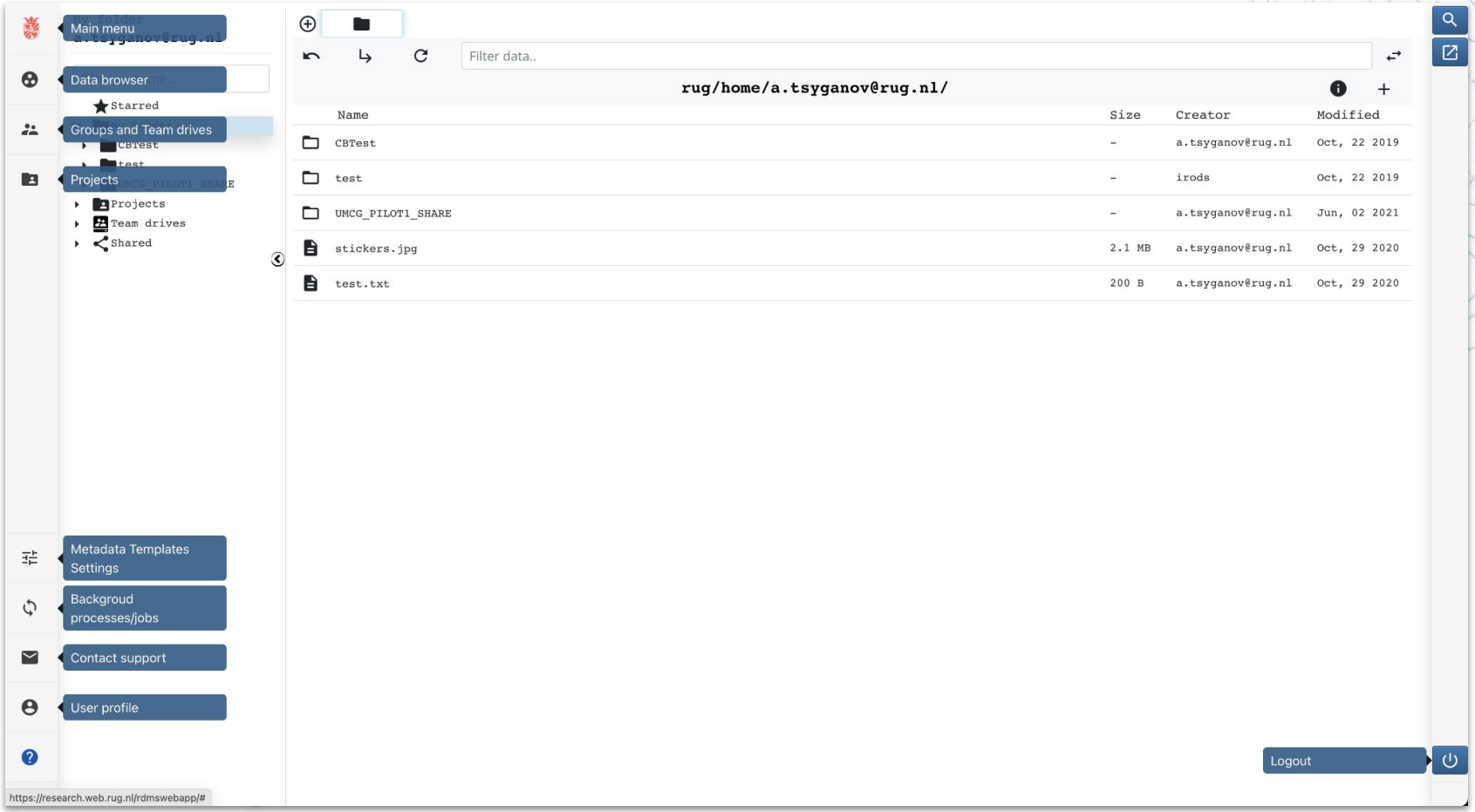
- Mutable lists of metadata (id's)
- Object definition: project/dataset etc..

Web Interface

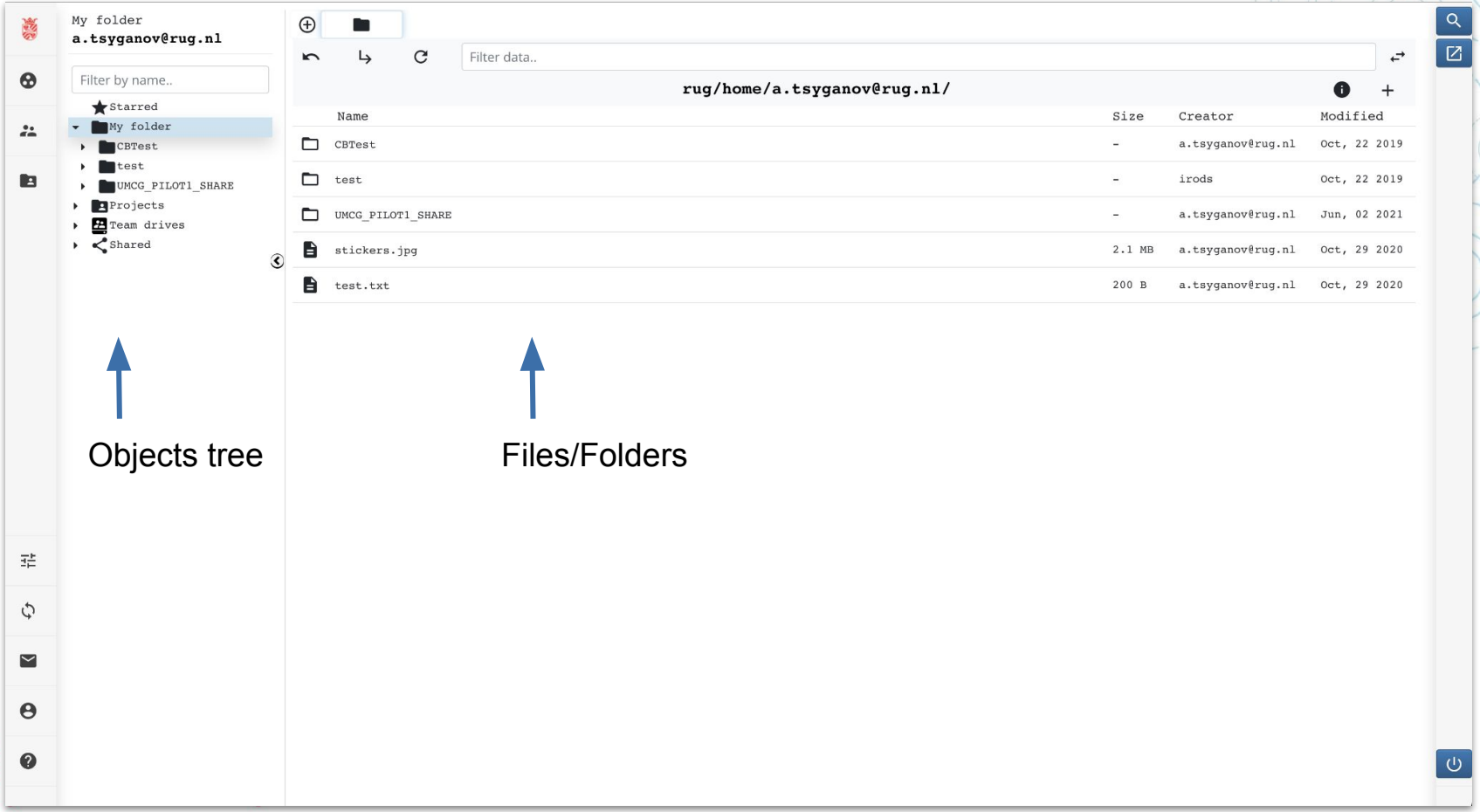
RUG RDMS: Web Interface



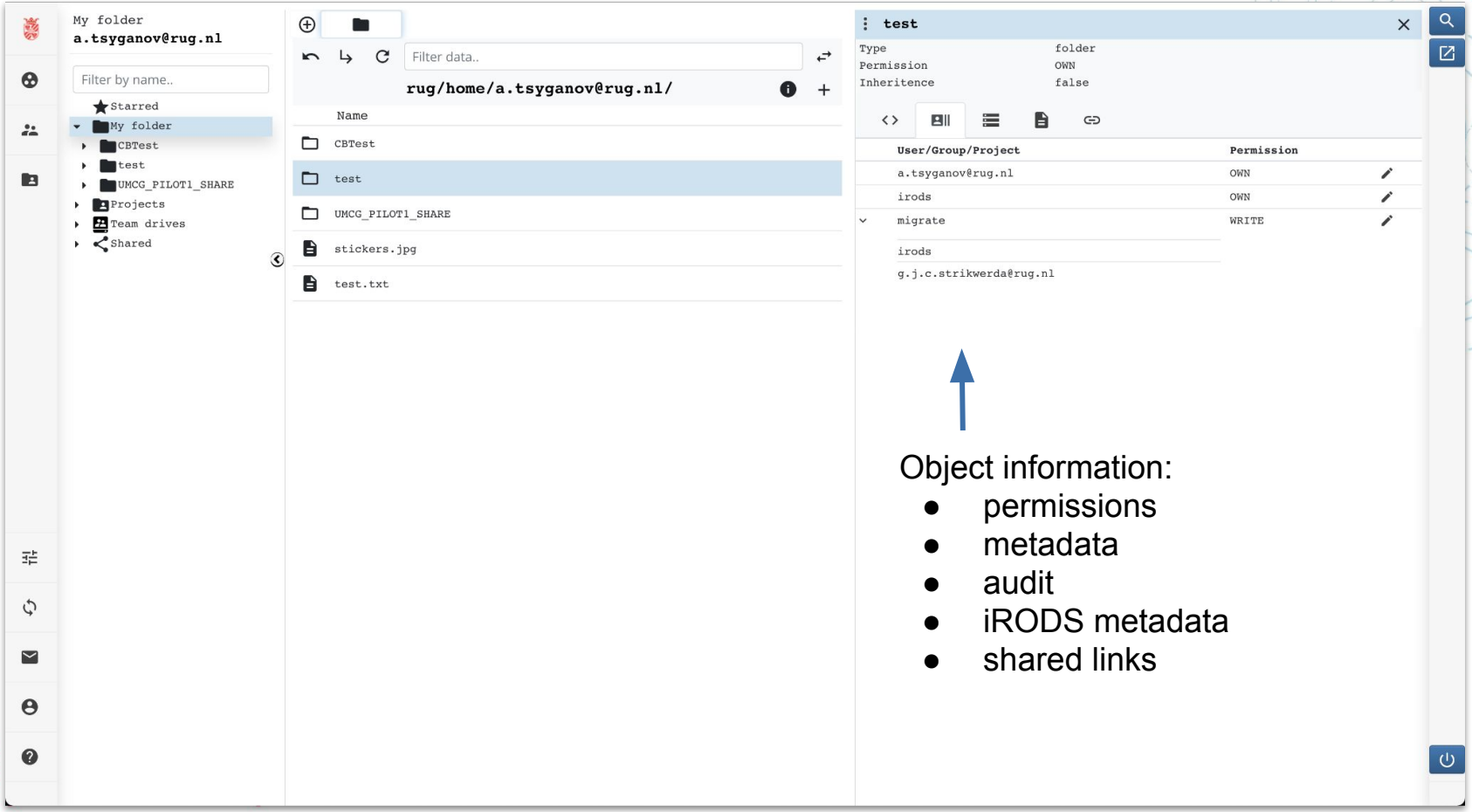
RUG RDMS: Web Interface



RUG RDMS: Web Interface



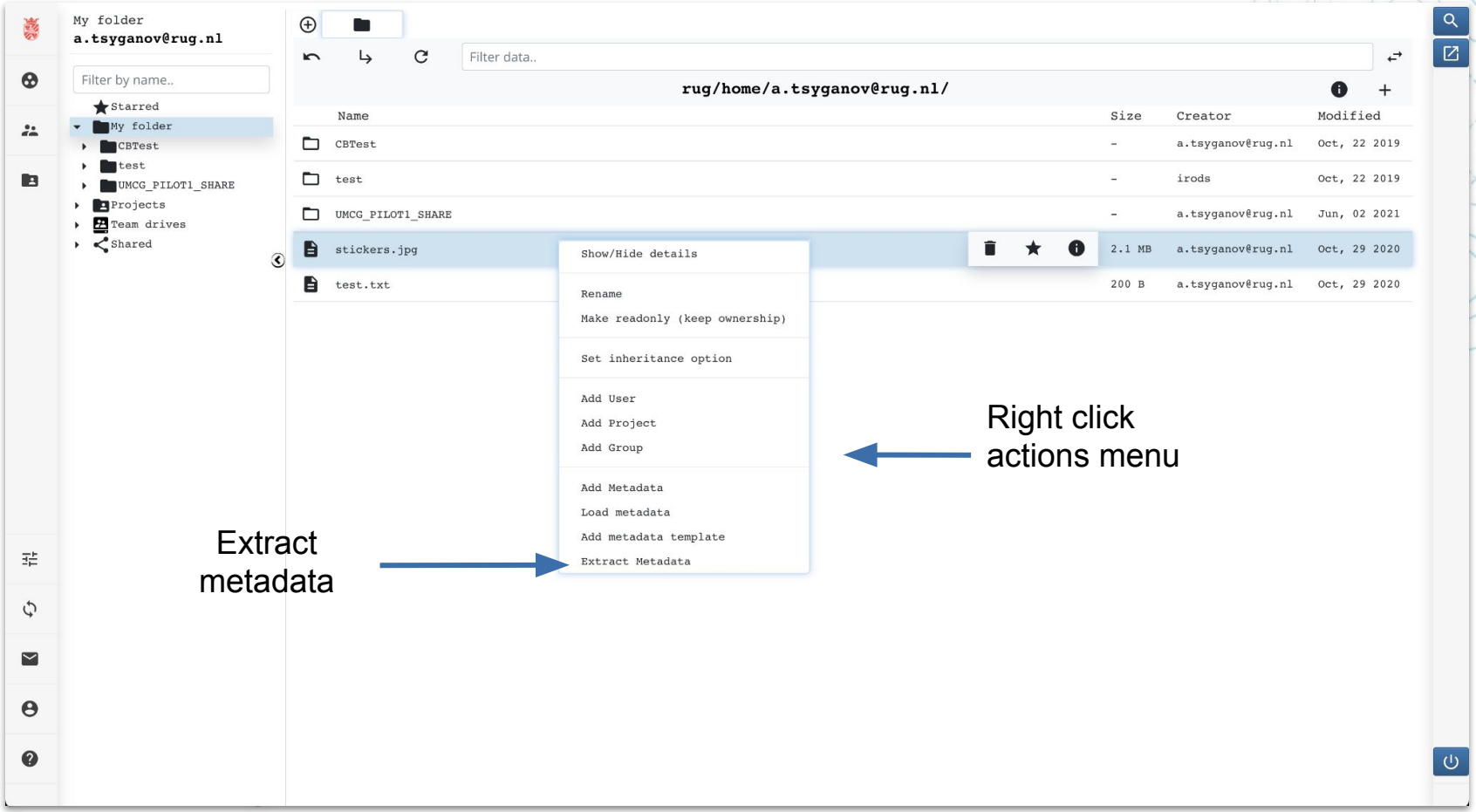
RUG RDMS: Web Interface



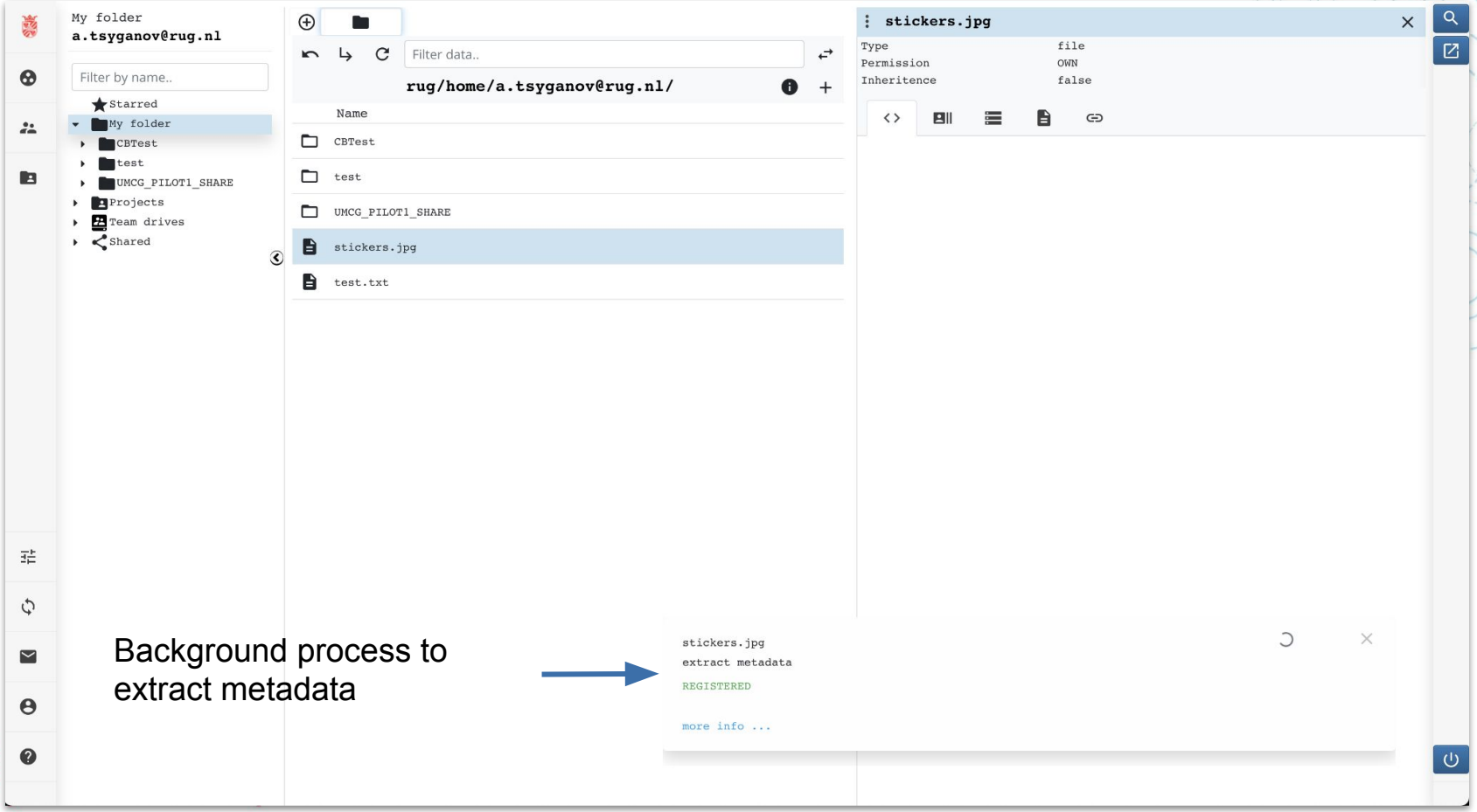
Object information:

- permissions
- metadata
- audit
- iRODS metadata
- shared links

RUG RDMS: Web Interface

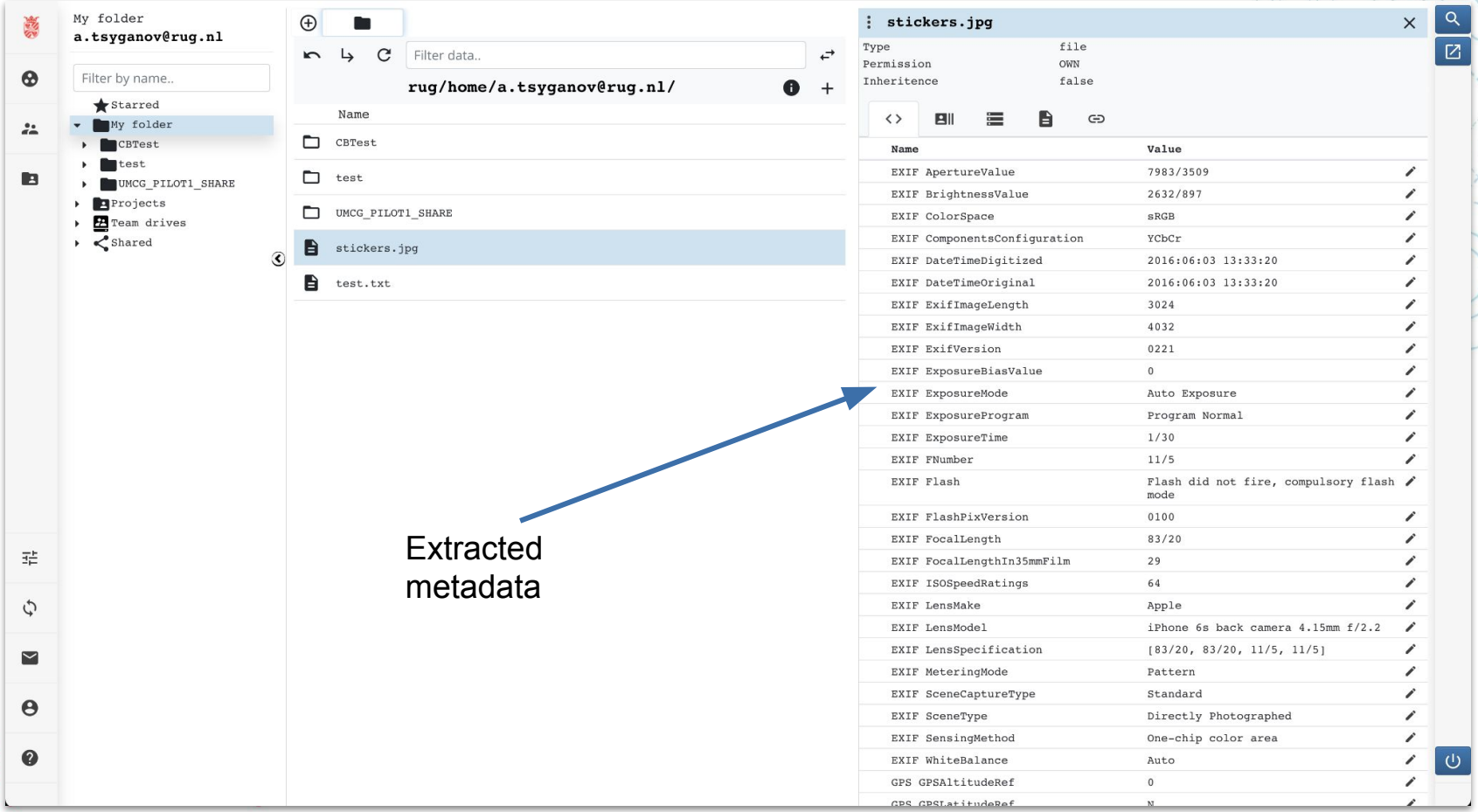


RUG RDMS: Web Interface



Background process to extract metadata

RUG RDMS: Web Interface



Extracted metadata

iRODS rules

RUG RDMS: iRODS rules

- Custom rules:
  - delayed rules and monitoring
  - copy data with permissions & metadata
  - automatic metadata extraction
  - custom policies engine
  - audit and permission control extra rules
- Rules to support tape library archiving

Custom policies engine

RUG RDMS: Custom policies engine

Data object: file or folder

RUG RDMS: Custom policies engine

Data object: file or folder

Special metadata as json

- Name:
  sysmdt_rdms_policy_<unique hash of the full path to object>
- Value:
  {"policy_name" : <policy_name>
  ,"policy_creator" : <policy_creator>
  ,"input_parameters" :
  {
  <parameter_name>: <parameter_value,
  <parameter_name>: <parameter_value,
  <parameter_name>: <parameter_value,
  ...
  }
  }
- Unit:
  POLICY|<policy type>|<policy status>|<policy user>

RUG RDMS: Custom policies engine

Data object: file or folder

RUG RDMS: Custom policies engine

Data object: file or folder

/testZone/home/Projects/project0_5n1 => one direction hash transformation =>
=> 2c7197f0a89e1c842180756537534a81a069be79e8ec6fa1473af21c

- Name:
  sysmdt_rdms_policy_2c7197f0a89e1c842180756537534a81a069be79e8ec6fa1473af21c
- Value:

```
{"policy_name"        : "project_user_participation_enddate",
 "policy_creator"     : "atsG",
 "input_parameters" :
                {      "user_name"  : "atsG",
                       "end_date"   : "10/06/2021 11:52",
                       "date_format" : "%d/%m/%Y %H:%M"
                }
}
```

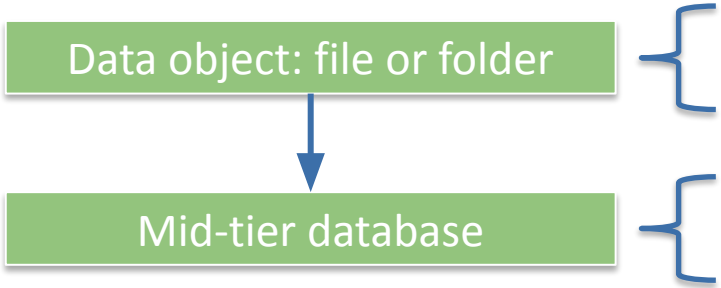
- Unit:
  POLICY|PROJECT|TORUN|atsG

RUG RDMS: Custom policies engine

Data object: file or folder

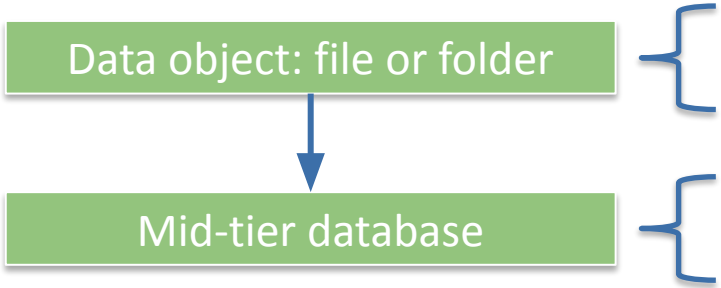
Policy metadata

RUG RDMS: Custom policies engine



university of groningen / center for information technology

RUG RDMS: Custom policies engine




```
rugrdmsdb=> \d RRREGISTERED_POLICIES_SYS
                    Table "public.rrregistered_policies_sys"
        Column          |            Type             | Collation | Nullable | Default
------------------------+-----------------------------+-----------+----------+---------
policy_id               | text                        |           |          |
policy_verification_id  | text                        |           |          |
policy_status           | character varying(255)      |           |          |
created                 | timestamp without time zone |           |          | now()
changed                 | timestamp without time zone |           |          | now()
policy_object           | text                        |           |          |
Indexes:
```

RUG RDMS: Custom policies engine

```
rugrdmsdb=> select policy_id, policy_verification_id, policy_object from RRREGISTERED_POLICIES_SYS;
                            policy_id                             |                policy_verification_id                |             policy_object
------------------------------------------------------------------+------------------------------------------------------+---------------------------------------
 sysmdt_rdms_policy_2c7197f0a89e1c842180756537534a81a069be79e8ec6fa1473af21c | 3529f650e5142fd86292fb3ffee692bd216db829ec47e1d5655d076a | /testZone/home/Projects/PROJECT0_8n
 sysmdt_rdms_policy_2c7197f0a89e1c842180756537534a81a069be79e8ec6fa1473af21c | ce16422bfa57b8fb3b9dc6a2c88e343aa492dc10c19cd8bf469a72a3 | /testZone/home/Projects/PROJECT0_8n_sub1
 sysmdt_rdms_policy_b2087ecaa307d420dcca5d80476f977fdda8bec0c2baeca9ae64adcb | 63dfc470b5f239238657f79db584fa43a94fe55255f16f0e10d6c3fe | /testZone/home/Projects/PROJECT0_8n
 sysmdt_rdms_policy_2c7197f0a89e1c842180756537534a81a069be79e8ec6fa1473af21c | 0f73476b83f74f1cbc06f51d7c791de2d059a72ca15e399909dfe493 | /testZone/home/Projects/project0_5n1
(4 rows)
```

key

verification hash

object name

- Code injection protection: if verification is turned on then only policies with the correct verification hash will be triggered.
- Protection from a manual change of the metadata for policies

RUG RDMS: Custom policies engine

Server cron job (as rods user)

RUG RDMS: Custom policies engine

Server cron job (as rods user)

icommands - get all metadata that are policies to run

RUG RDMS: Custom policies engine

Server cron job (as rods user)

icommands - get all metadata that are policies to run

get policy information:
- metadata id
- user that needs to run policy

RUG RDMS: Custom policies engine

Server cron job (as rods user)

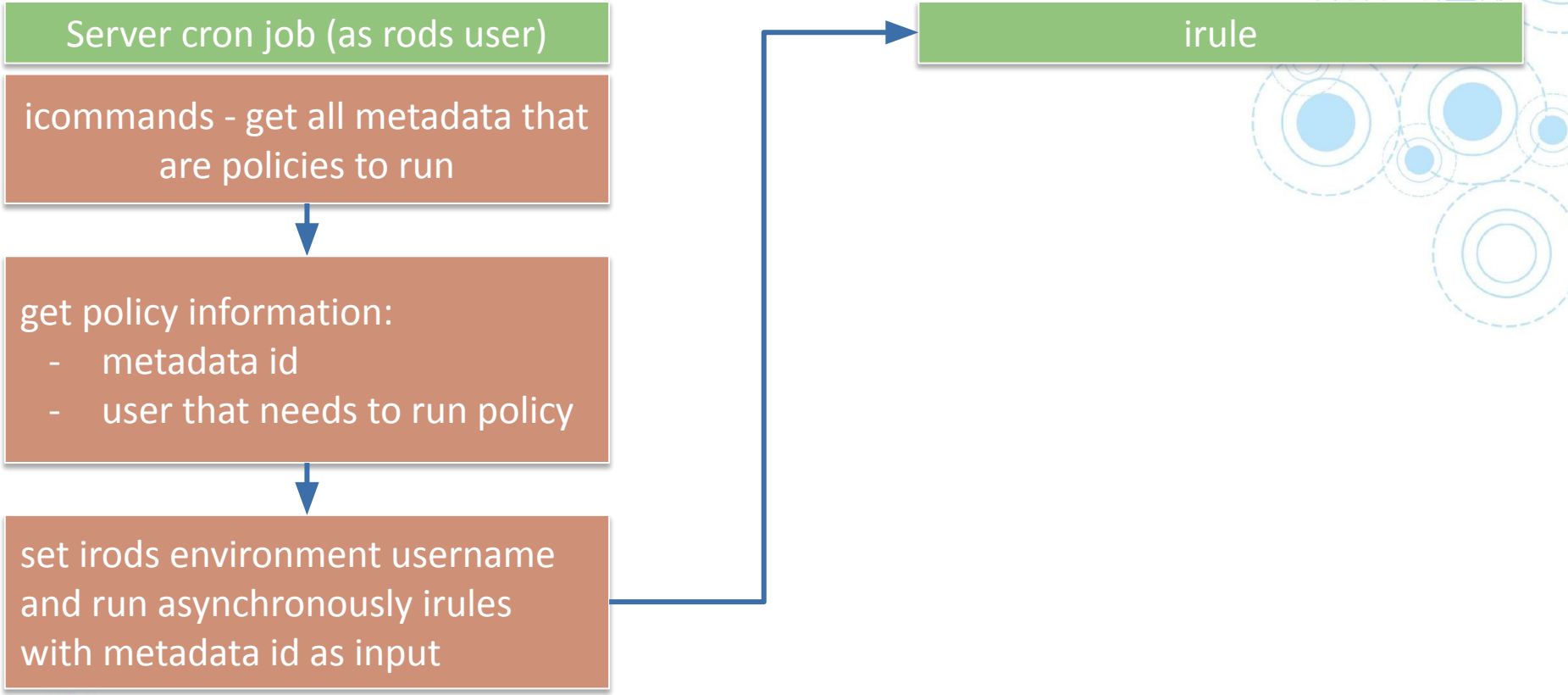
icommands - get all metadata that are policies to run

get policy information:
- metadata id
- user that needs to run policy

set irods environment username and run asynchronously irules with metadata id as input

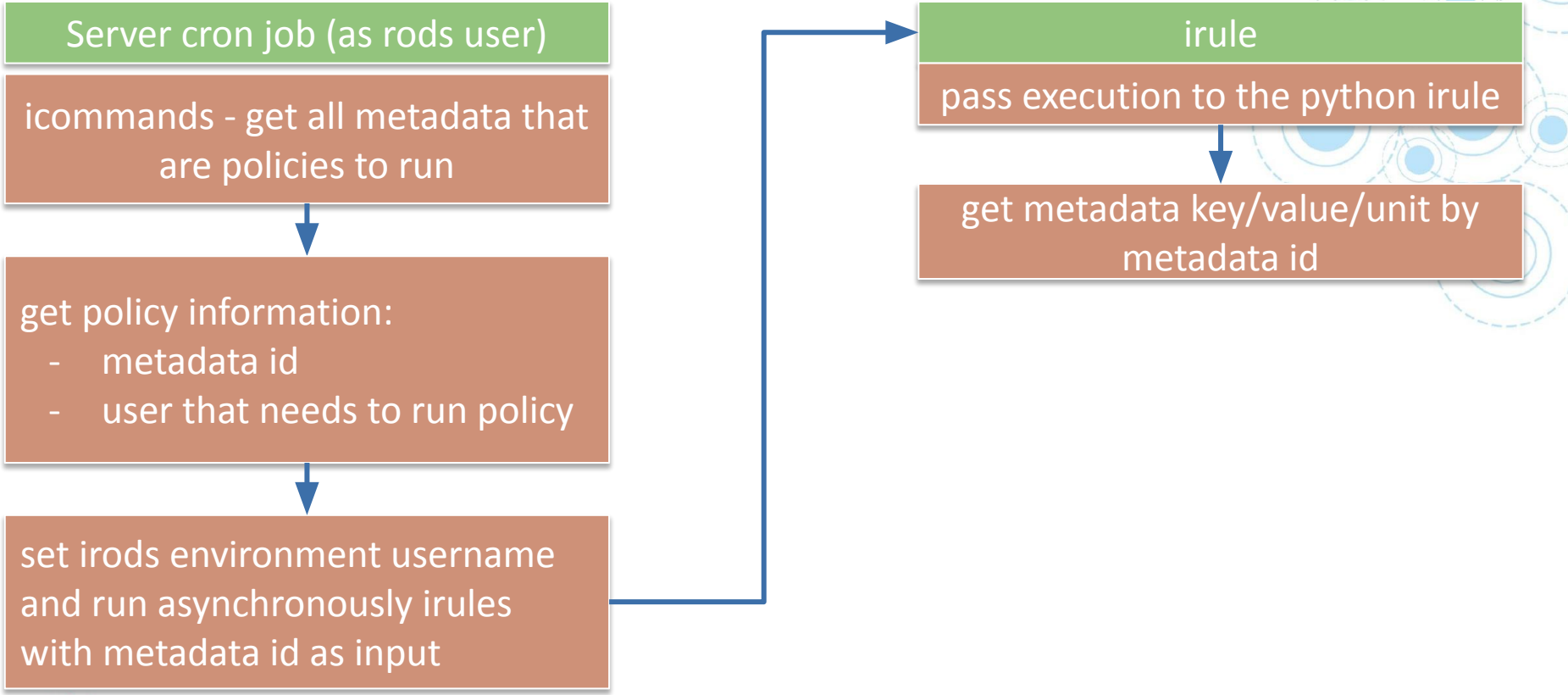
university of groningen / center for information technology

RUG RDMS: Custom policies engine



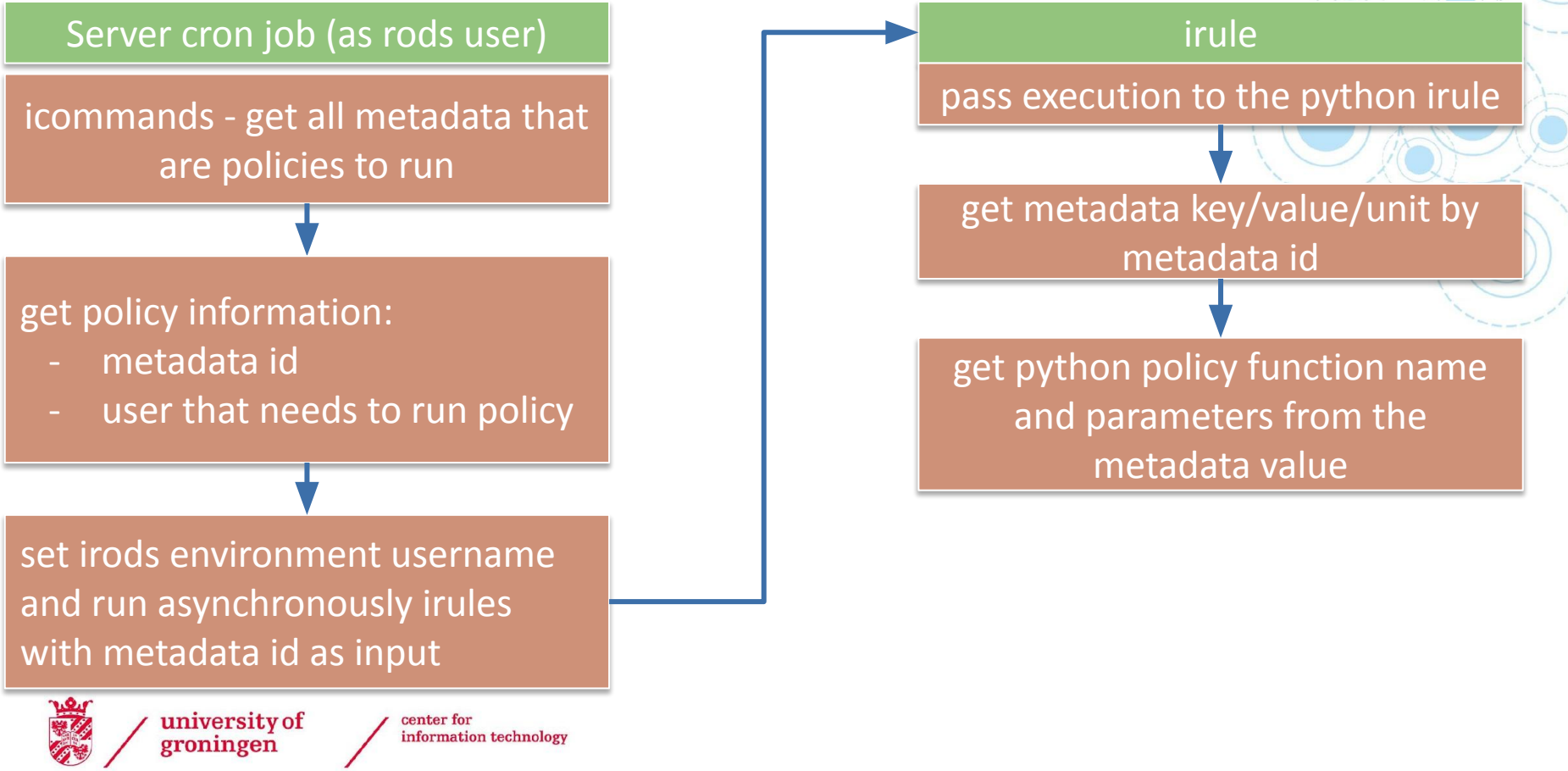
university of groningen / center for information technology

RUG RDMS: Custom policies engine



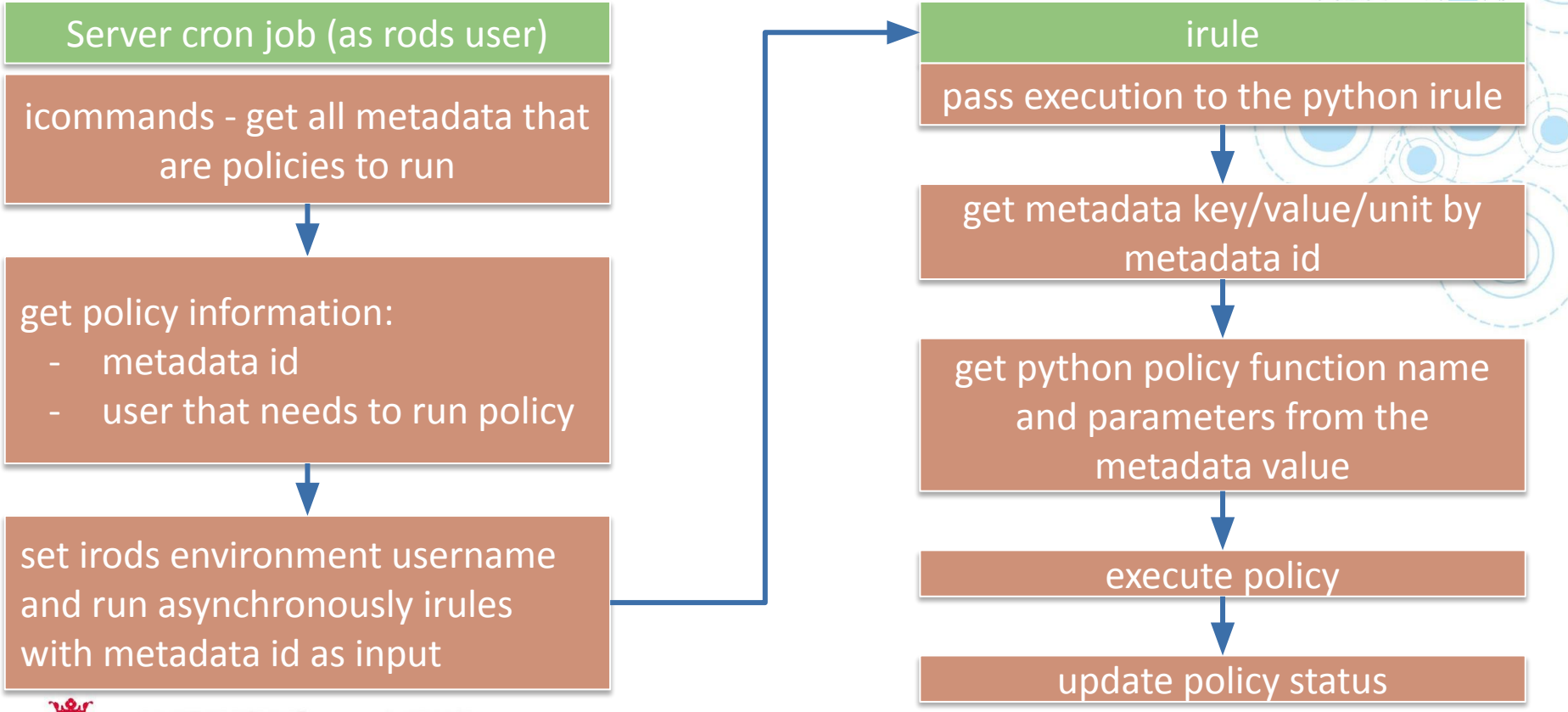
university of groningen / center for information technology

RUG RDMS: Custom policies engine



university of groningen / center for information technology

RUG RDMS: Custom policies engine



university of groningen / center for information technology

RUG RDMS: Custom policies engine

```
from .policy_project_user_participation_enddate  import run_policy_project_user_participation_enddate


 C_AVAILABLE_POLICIES = {
     "project_user_participation_enddate" : run_policy_project_user_participation_enddate
 }



# method to run policy that we've fetched from the metadata of the object
def run_policy(self):
   l_function_name, l_parameters =  self.parce_policy() # parce json
   if self.policyIsValid():
       if self.c namespace.C AVAILABLE POLICIES.has key(l function name):
           self.c_namespace.C_AVAILABLE_POLICIES[l_function_name]( self, l_parameters)  # execute code
```

Metadata Templates

RUG RDMS: Metadata Templates

Why our approach?

- Metadata is the necessary component to transform *data* into *knowledge*.
- A well-written metadata template is *crucial* to tag the data being stored
- Several *requests* for metadata template in different research domains:
  - social sciences, archeology, microbiology

RUG RDMS: Metadata Templates

```
{
   "Publication":
       {
           "Title": "An overview of RDMS",
           "Authors": [
                  "RDMS Team Member 1",
                  "RDMS  Team Member 2" ],
           "Date": "2021",
           "Publisher": "RUG",
           "Pages" : "4"
       }
}
```

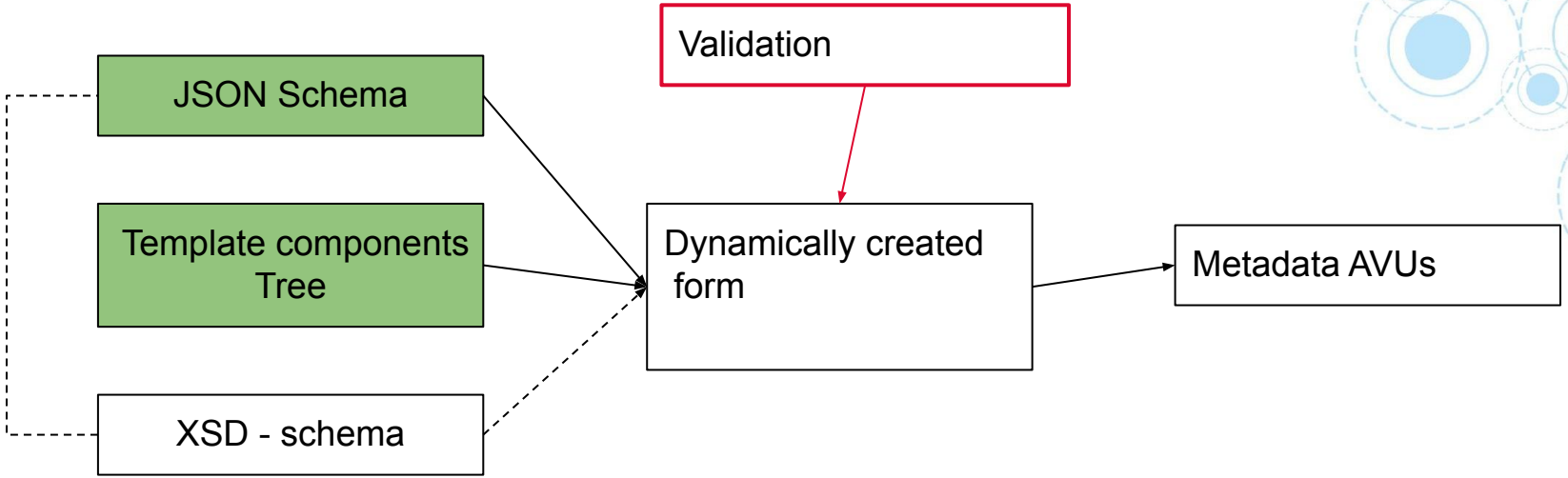
```
{
   "Recording": {
       "Camera": "Ladybug5+",
       "Model" : "LD5P-U3-51S5C-R",
       "Subject": "Asfalt quality",
       "Date": "2021",
       "Azimuth angle": 140
   }
}
```

RUG RDMS: Metadata Templates

- There is no solution of *one-size-fits-all*
  - Each **research domain** has **specific** metadata-attributes
  - Within the same domain **research questions** may have different collection of tags
  - The technical skills of the users are very different

Provide the *User* with the possibility to *define* domain/research *specific* metadata templates

RUG RDMS: Metadata Templates - Technical solution



RUG RDMS: Metadata Templates - Technical solution

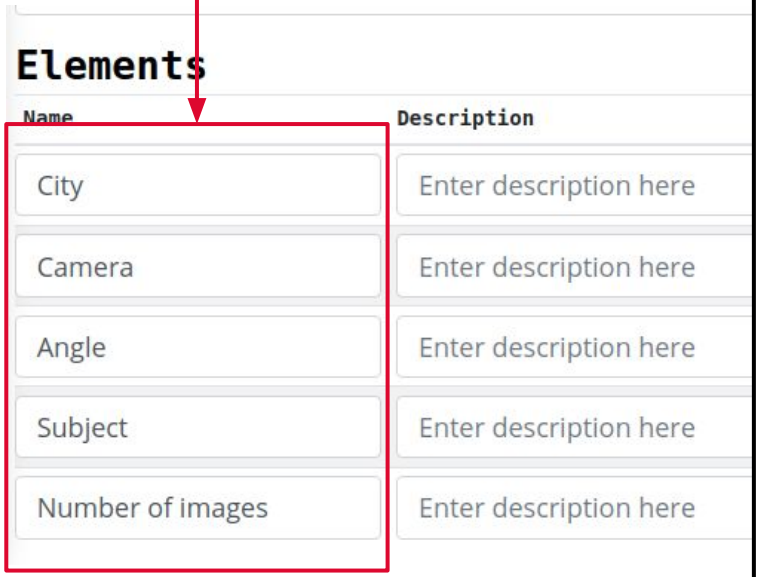
Self defined structure

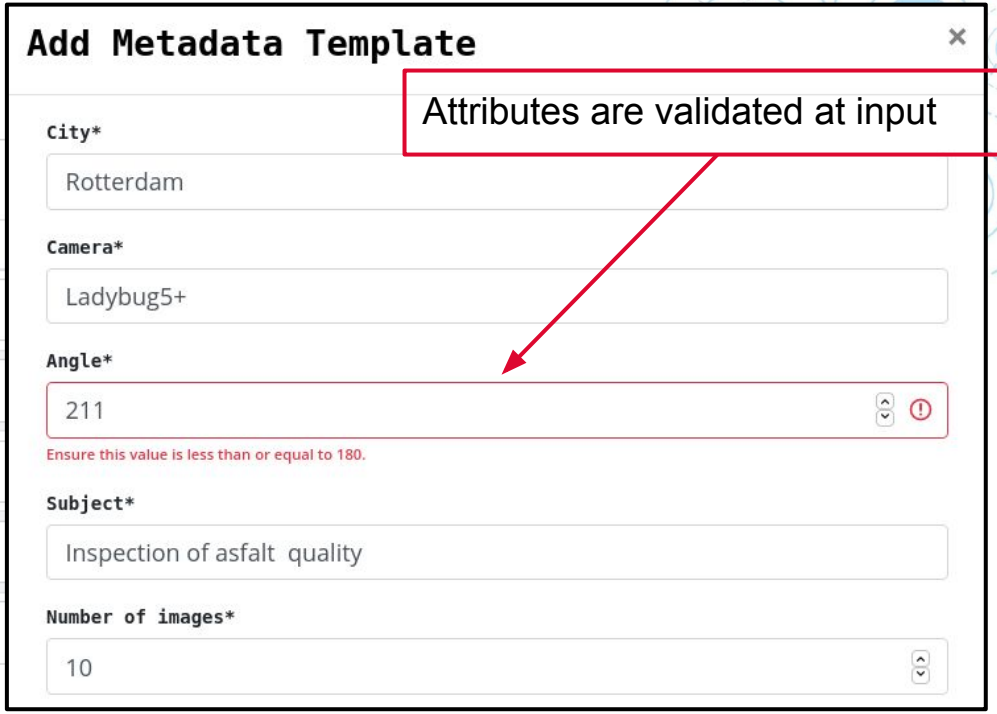
Self defined types

university of groningen / center for information technology

RUG RDMS: Metadata Templates - Technical solution

Self defined structure





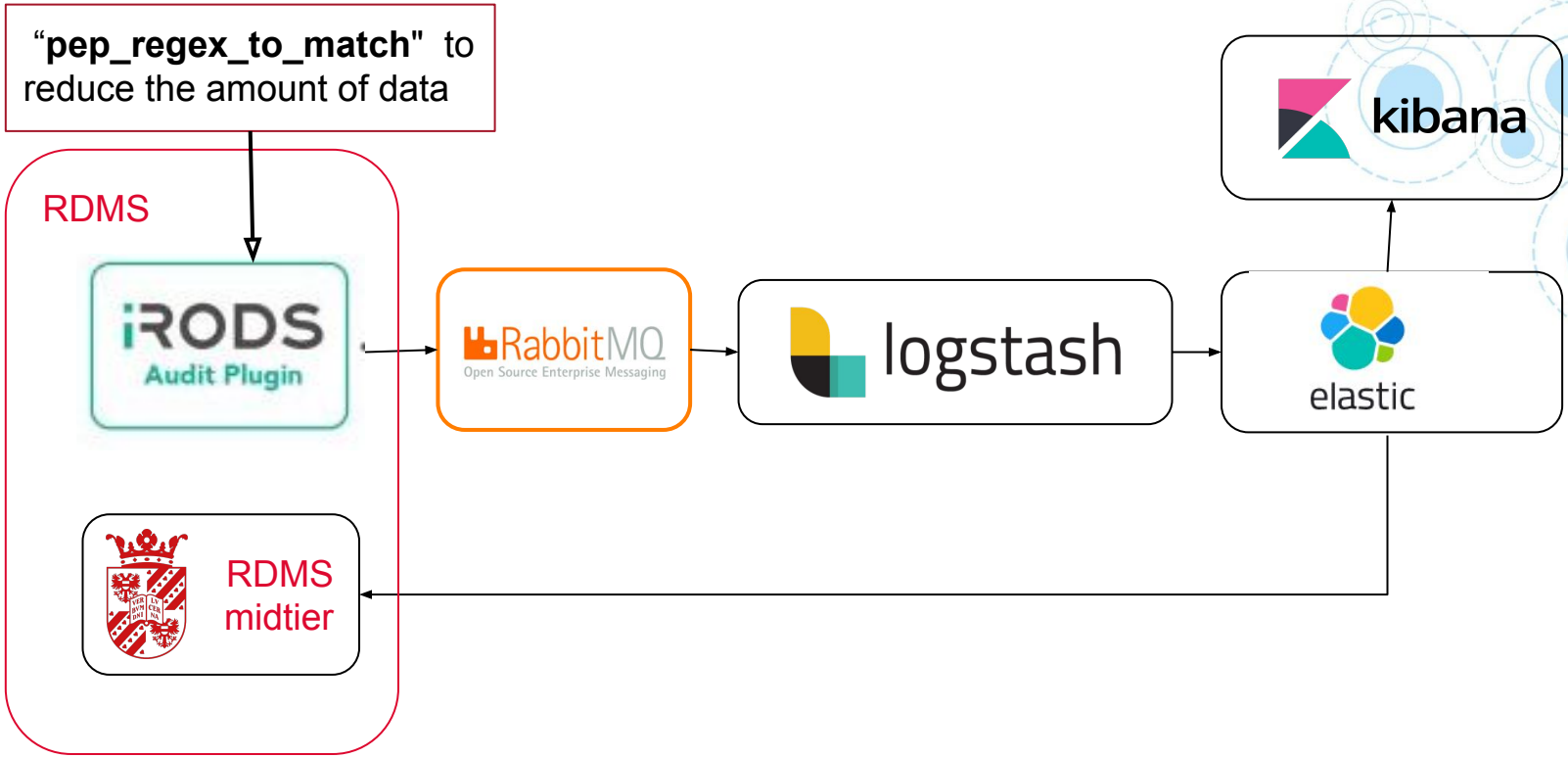
university of groningen / center for information technology

Auditing

RUG RDMS: Auditing

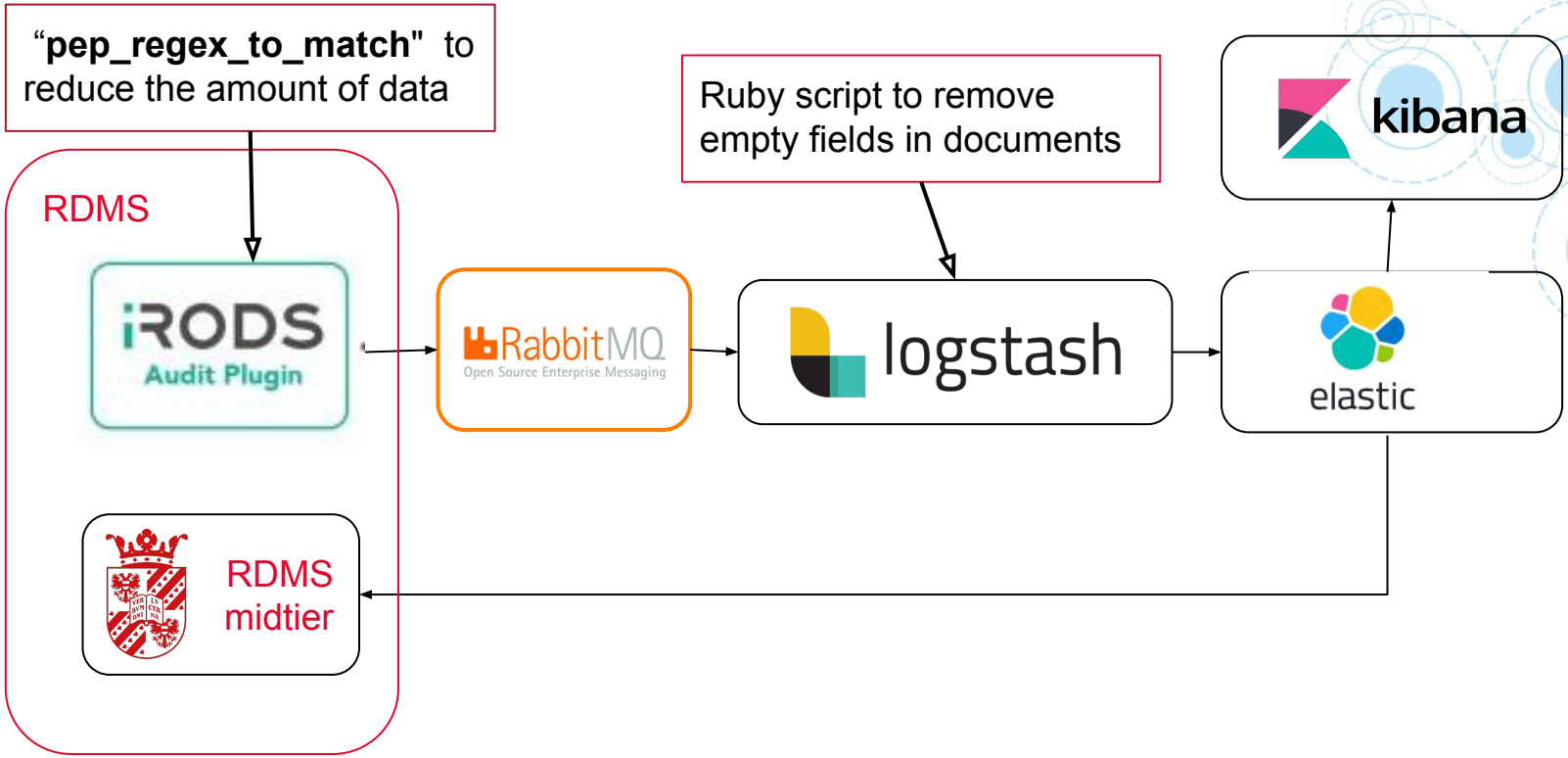
university of groningen / center for information technology

RUG RDMS: Auditing



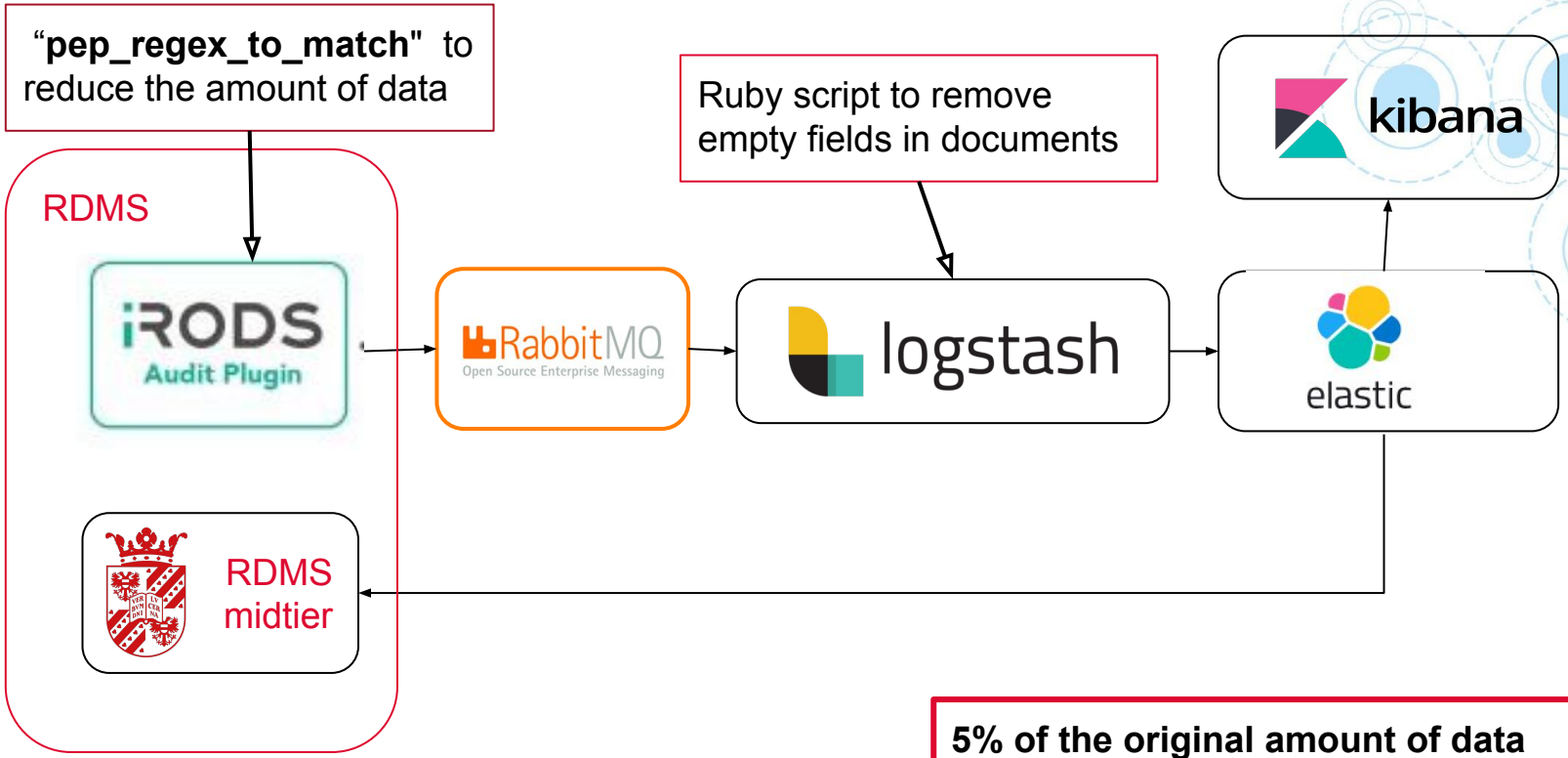
university of groningen / center for information technology

RUG RDMS: Auditing



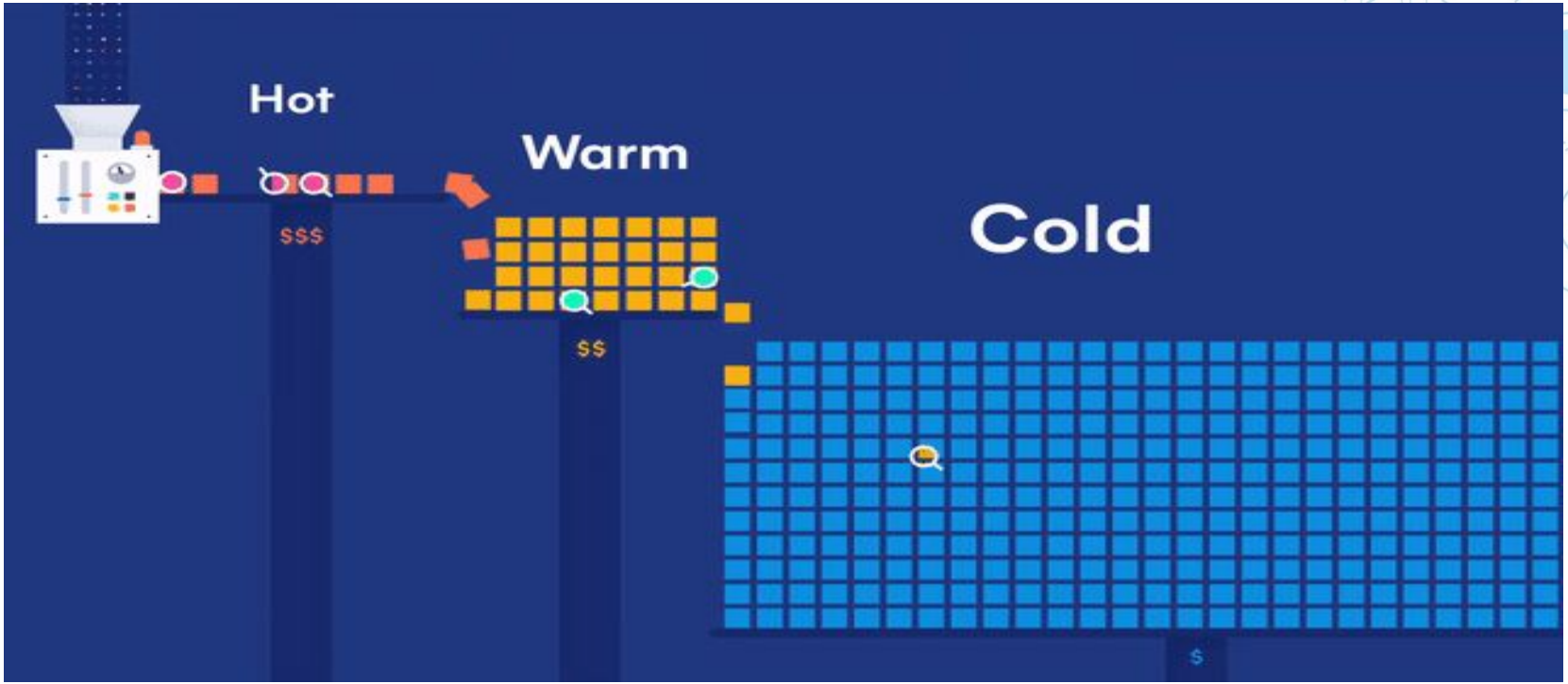
university of groningen / center for information technology

RUG RDMS: Auditing



university of groningen / center for information technology

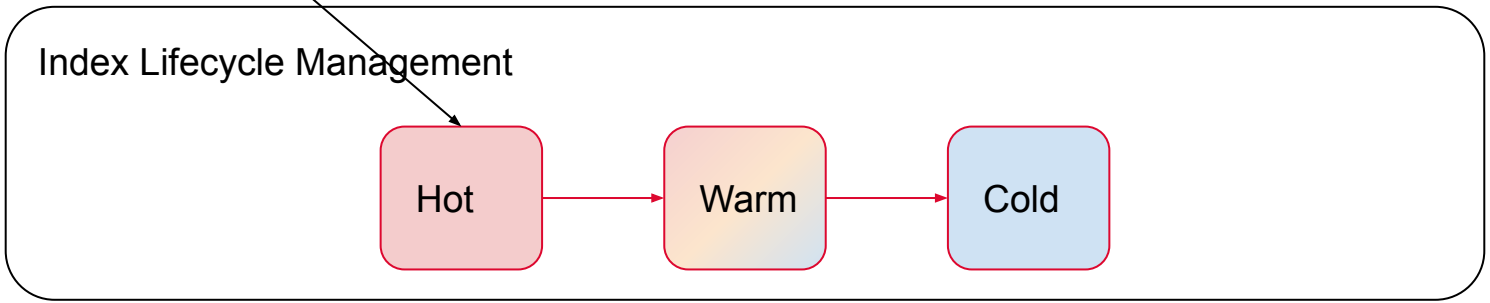
RUG RDMS: Auditing - Index Lifecycle Management



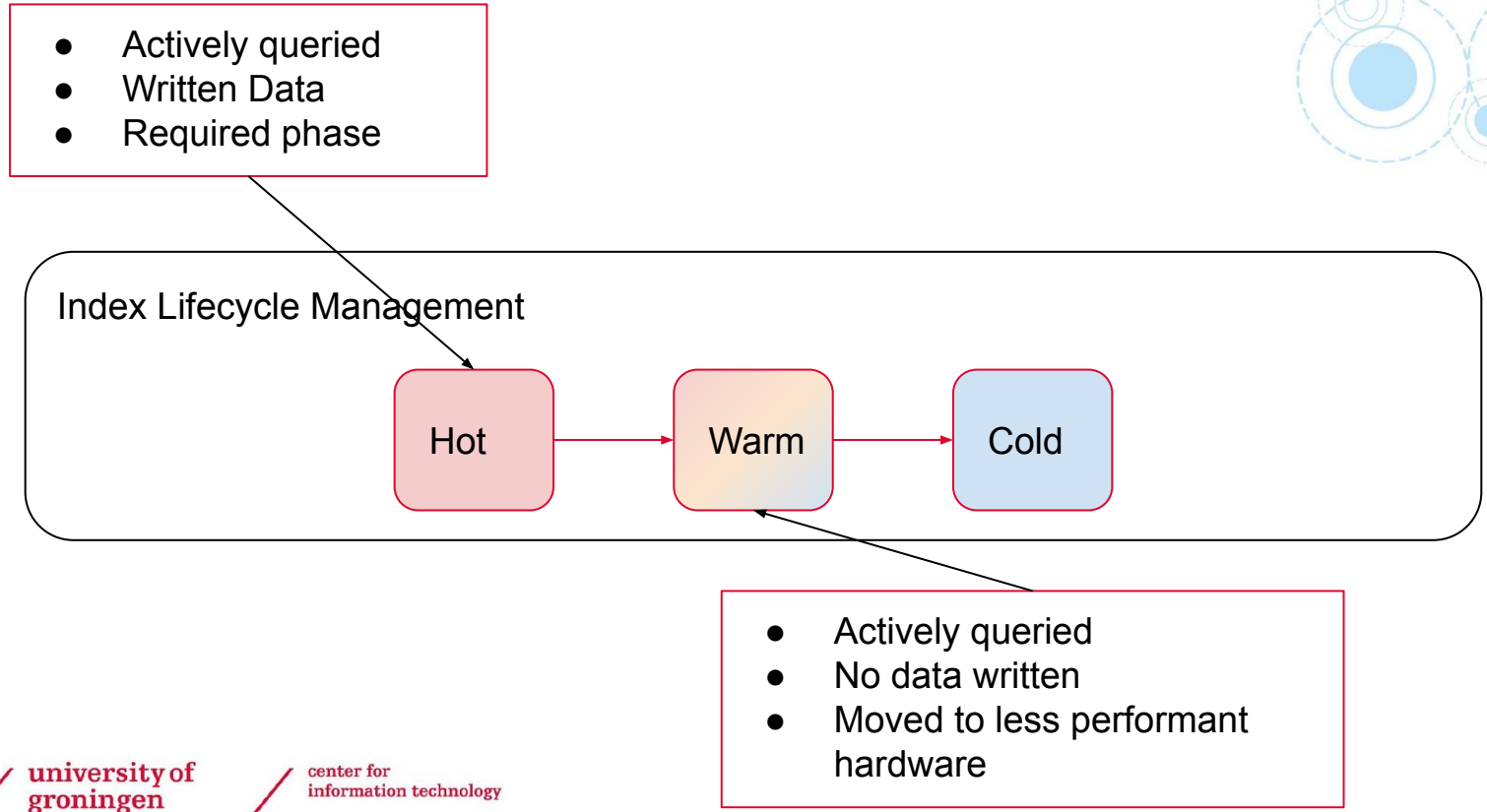
university of groningen / center for information technology

Elastic Search - Index Lifecycle Management

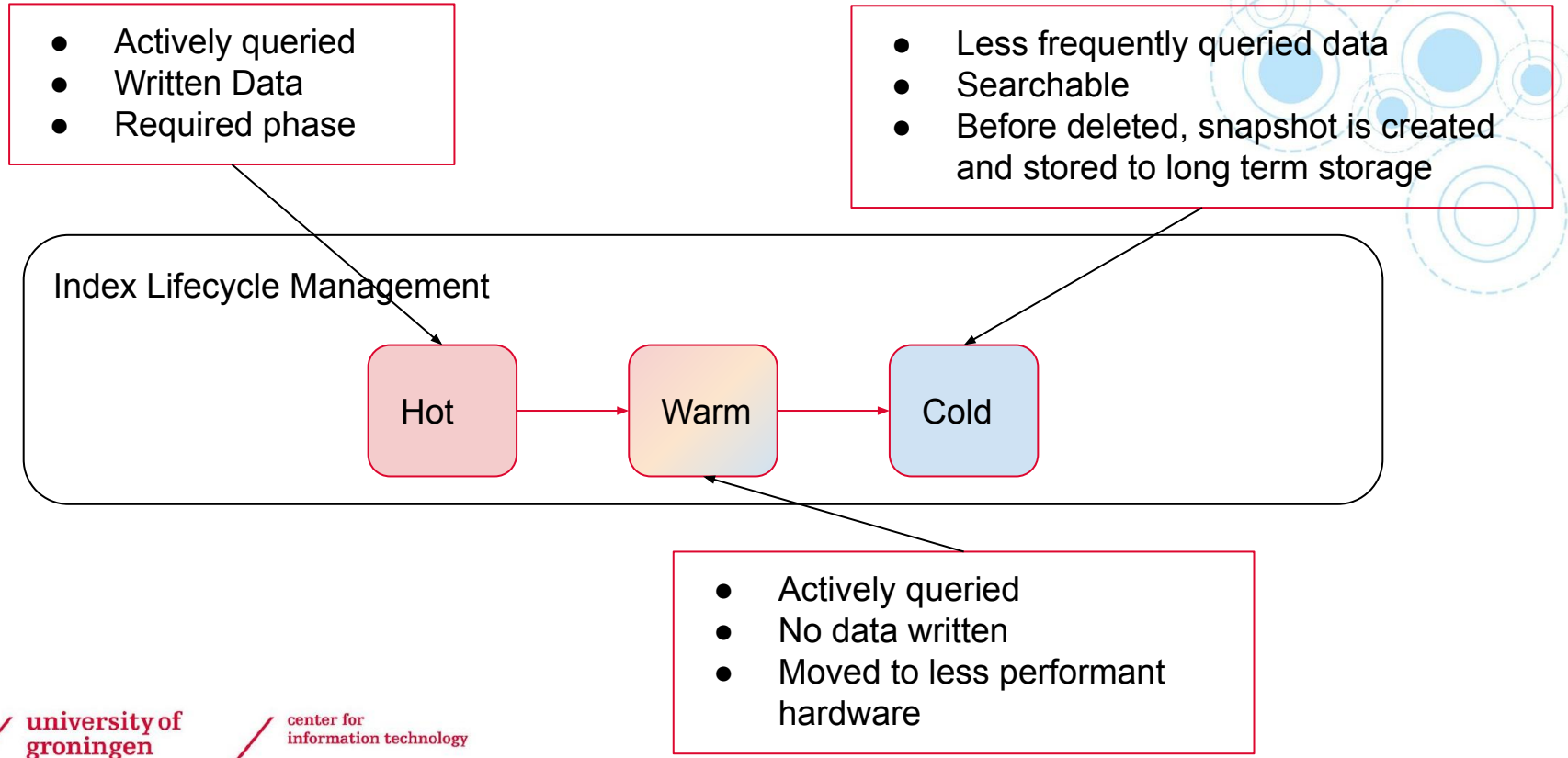
- Actively queried
- Written Data
- Required phase



Elastic Search - Index Lifecycle Management



Elastic Search - Index Lifecycle Management



RUG RDMS: What's next?

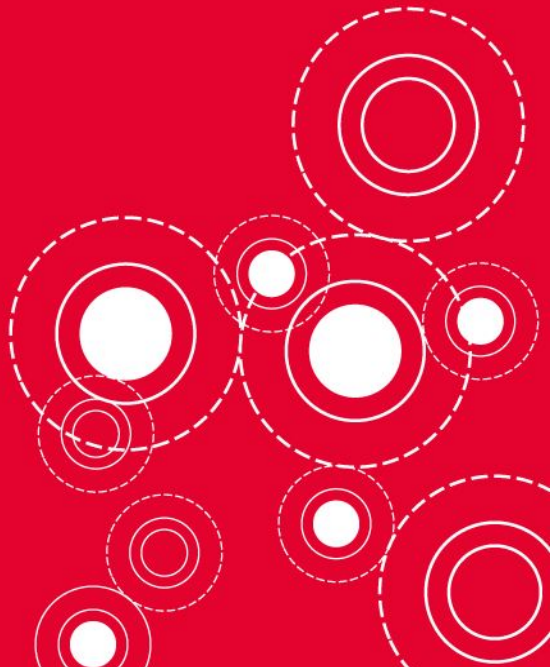
- Make code open source & improve deployment
- Integrate switch between different iRODS zones
- Finish customers use cases (currently there are 7 major project running from different faculties)
- Make iRODS NFS mount
- File level versioning
- Multi factor authentication & External users
- Fail2ban
- Windows ingest large files
- Add metadata via mount
- Next version of the metadata templates
- Auditing - SLM snapshots

university of groningen / center for information technology

CIT

rdms-support@rug.nl

Q&A?



rug.nl/cit