IVOA Provenance DM in TAP :

Issues and solutions

F.Bonnarel, CDS
on behalf of M.Servillat, M.Louys, M.Nullmeier, M.Sanguillon, L.Michel

CENTRE DE DONNÉES ASTRONOMIQUES DE STRASBOURG

ESCAPE
European Science Cluster of Astronomy & Particle physics ESFRI research Infrastructures

IVOA

Why Provenance in TAP ?

- Provenance information can be attached to data in various ways :
  - Embedded in the data « header » itself
  - Linked to the data record via DataLink or URL
  - Retrievable via ProvSAP via data id.
- In addition to that , Provenance metadata in a TAP service will allow to discover « data » by constraining Provenance features.
  - It's a « reverse » mechanism.

« The » issue = complexity

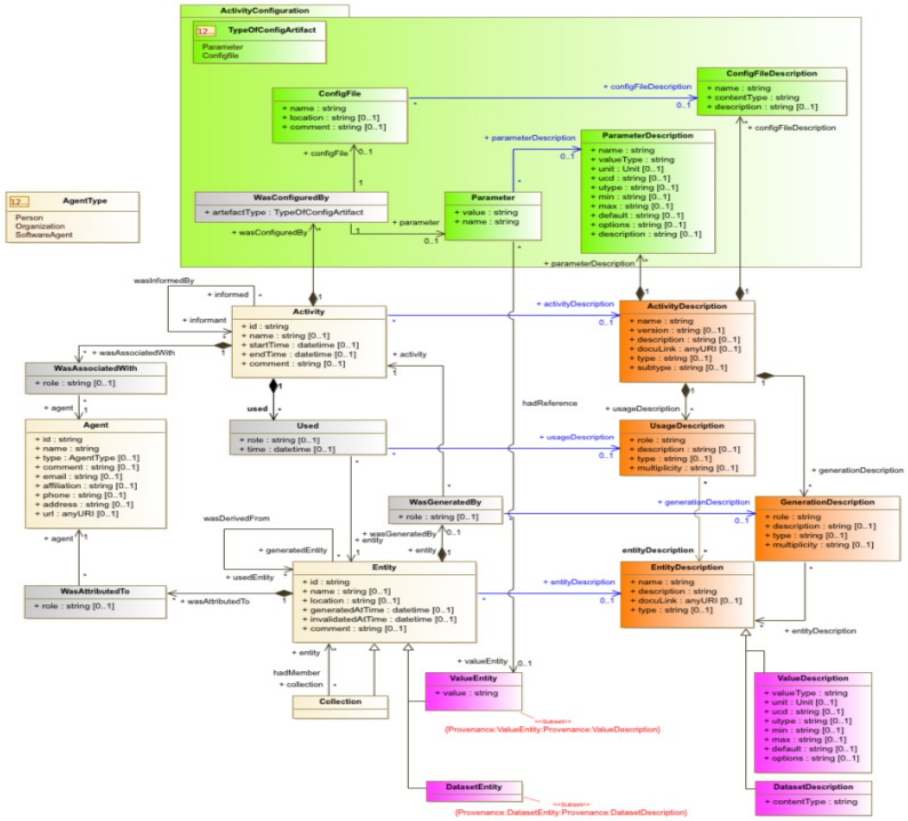


Figure 8: Full class diagram of the IVOA Provenance Data Model.

« The » issue = complexity

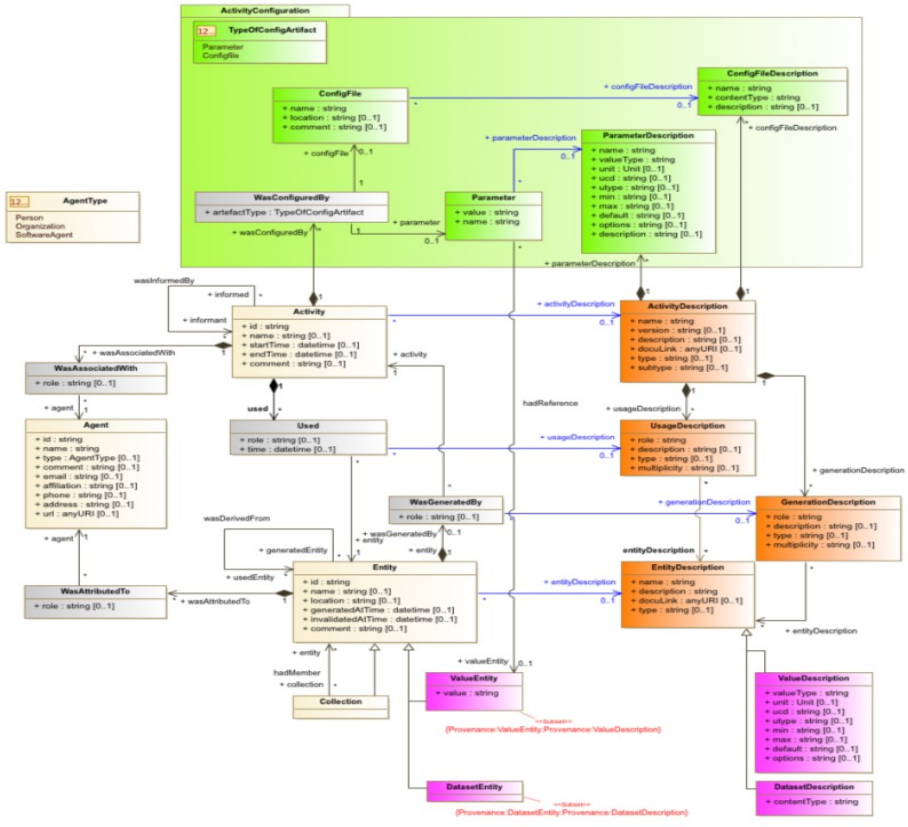


Figure 8: Full class diagram of the IVOA Provenance Data Model.

1 table per class ?

---→ ProvTAP

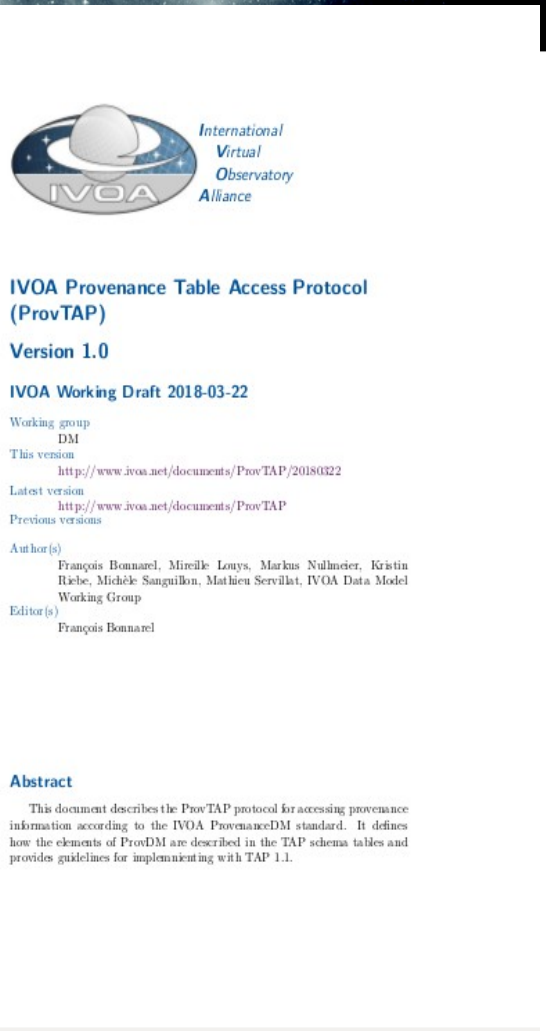
ProvTAP status

- There is an internal draft
on the IVOA DAL page

- TAP schema mapping classes
as tables

- ProvHiPS (provenance of
HiPS and HiPS tiles) is
an implementation prototype

- From now examples and demos
from ProvHiPS



International Virtual Observatory Alliance

IVOA Provenance Table Access Protocol (ProvTAP)

Version 1.0

IVOA Working Draft 2018-03-22

Working group
DM
This version
http://www.ivoa.net/documents/ProvTAP/20180322
Latest version
http://www.ivoa.net/documents/ProvTAP
Previous versions

Author(s)
François Bonnarel, Mireille Louys, Markus Nullmeier, Kristin Riebe, Michèle Sanguillon, Mathieu Servillat, IVOA Data Model Working Group
Editor(s)
François Bonnarel

Abstract

This document describes the ProvTAP protocol for accessing provenance information according to the IVOA ProvenanceDM standard. It defines how the elements of ProvDM are described in the TAP schema tables and provides guidelines for implemnienting with TAP 1.1.

ProvTAP TAP_SCHEMA: Entity table

Name	ucd	utype	datatype	status
e_id	meta.id	voprov:Entity.id	char	M
e_name	meta.title	voprov:Entity.name	char	O
e_type	meta.code.class	voprov:Entity.type	char	O
e_rights	meta.code.class	voprov:Entity.rights	char	O
e_location	meta.ref.url	voprov:Entity.location	char	O
e_generated	time.start	voprov:Entity.generatedAtTime	char	O
e_invalidated	time.stop	voprov:Entity.invalidatedAtTime	char	O
e_comment	meta.description	voprov:Entity.comment	char	O
e_classtype	meta.code.class	voprov:Entity.classtype	char OPTION	M
e_value	stat.value	voprov:Entity.value	char	O
→ e_description	meta.id	voprov:Entity.description_id	reference	O

Table 2: Column description for Entity table. The e_classtype column may have the following two values :"dataset" and "value"

ProvTAP TAP_SCHEMA: parameterDescription table

Name	ucd	utype	datatype
pd_activitydescription	meta.id	voprov:ParameterDescription. activityDescription_id	char
pd_id	meta.id	voprov:ParameterDescription.id	char
pd_name	meta.title	voprov:ParameterDescription.name	param dependent
pd_description	meta.description	voprov:ParameterDescription.description	char
pd_datatype	meta	voprov:ParameterDescription.datatype	char
pd_unit	meta.unit	voprov:ParameterDescription.unit	char
pd_ucd	meta.ucd	voprov:ParameterDescription.ucd	char
pd_utype	meta	voprov:ParameterDescription.utype	char
pd_min	stat.min	voprov:ParameterDescription.min	param dependent
pd_max	stat.max	voprov:ParameterDescription.max	param dependent
pd_options	meta	voprov:ParameterDescription.options	param dependent

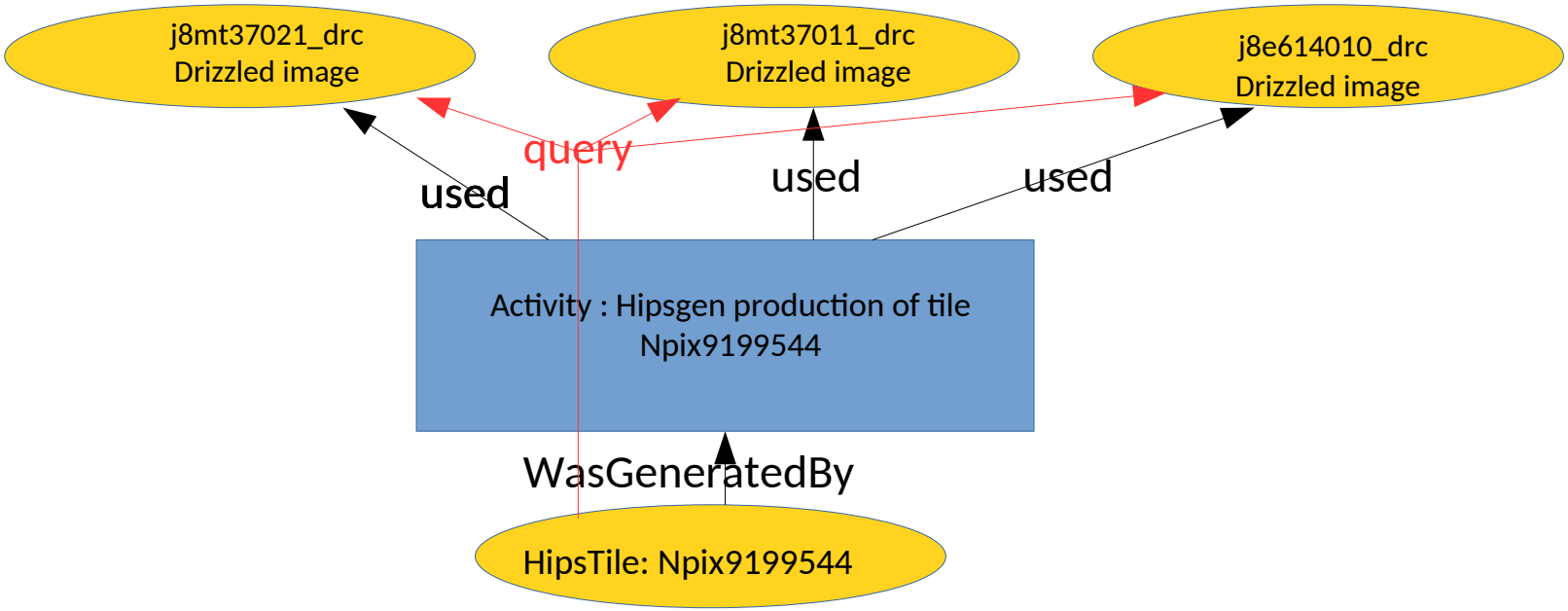
ProvTAP TAP_SCHEMA: parameter table

Name	ucd	utype	datatype
p_activity	meta.id	voprov:Parameter.activity_id	char
p_id	meta.id	voprov:Parameter.id	char
p_value	stat.value	voprov:Parameter.value	param dependent
→ p_description	meta.id	voprov:Parameter.parameterDescription_id	reference to parameter description

Table 8: Column description for Parameter table

ProvHiPS ADQL query examples :

Finding out drizzled images « progenitors » of a specific HiPS tile.



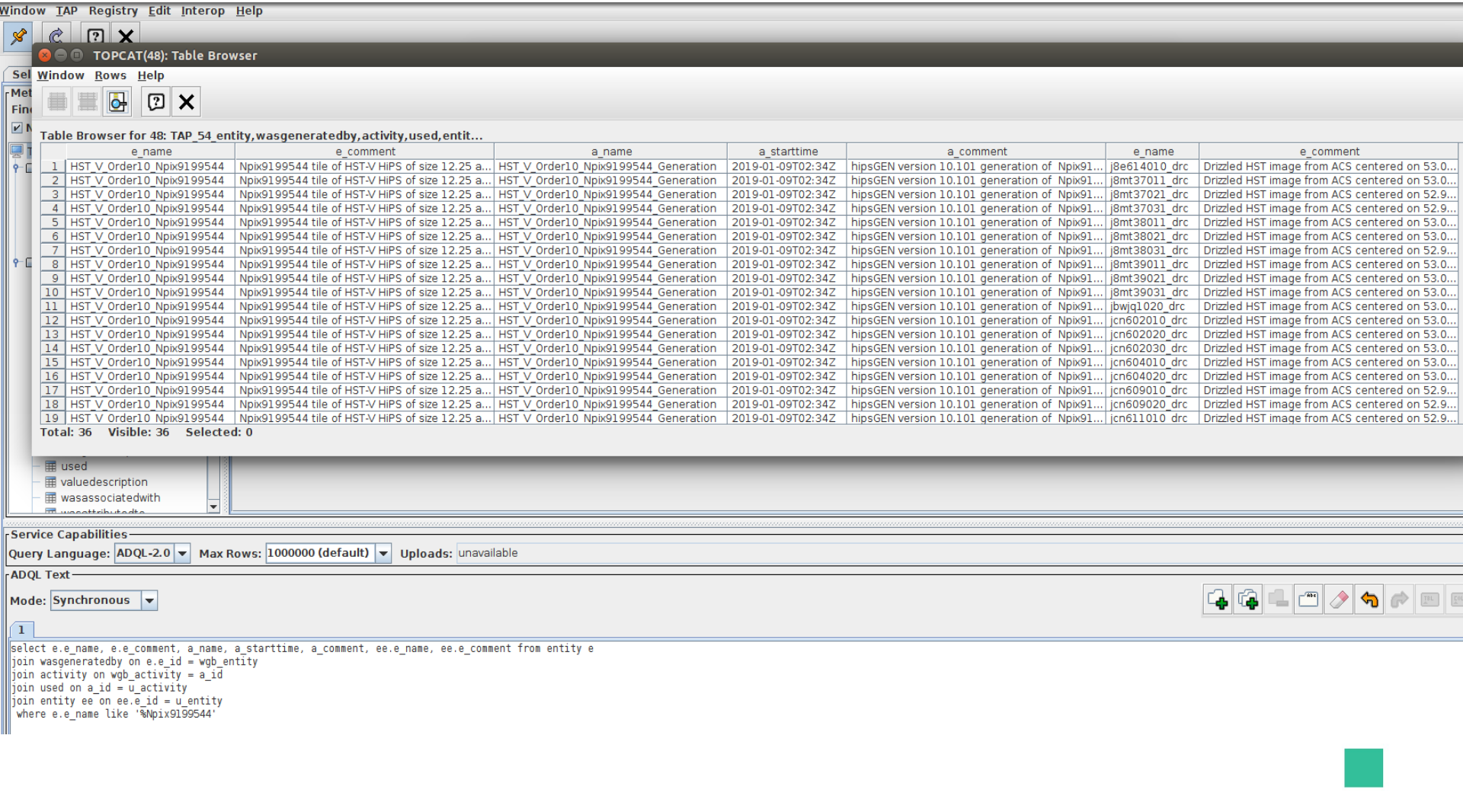
ProvHiPS ADQL query examples :

Finding out drizzled images « progenitors » of a specific HiPS tile.

```
select e.e_name, e.e_comment, a_name, a_starttime, a_comment, ee.e_name
    ee.e_comment from entity e
join wasgeneratedby on e.e_id = wgb_entity
join activity on wgb_activity = a_id
join used on a_id = u_activity
join entity ee on ee.e_id = u_entity
where e.e_name like '%Npix9199544'
```

ProvHiPS ADQL query examples :

Finding out drizzled images « progenitors » of a specific HiPS tile.



Issues

- Table is denormalized : a lot of redundant information
- Loop issue : several occurances of the same triplet (name,utype,ucd) in the same table for different « objects »
- Let’s try minimum or last step provenance by creating a standardized view

Solutions
-1 Single step = single table (= join)

- The join is a permanent view described in the TAP schema
- Columns :

  entity_name, entity_location, entity_comment, …

  generating_activity_name, generating_activity_starttime, ….

  agent_role, agent_name, ….

  used_entity_list
- → Redundancy may be avoided if we group all used entities ids in a single cell
- → possible Recursivity

Solutions

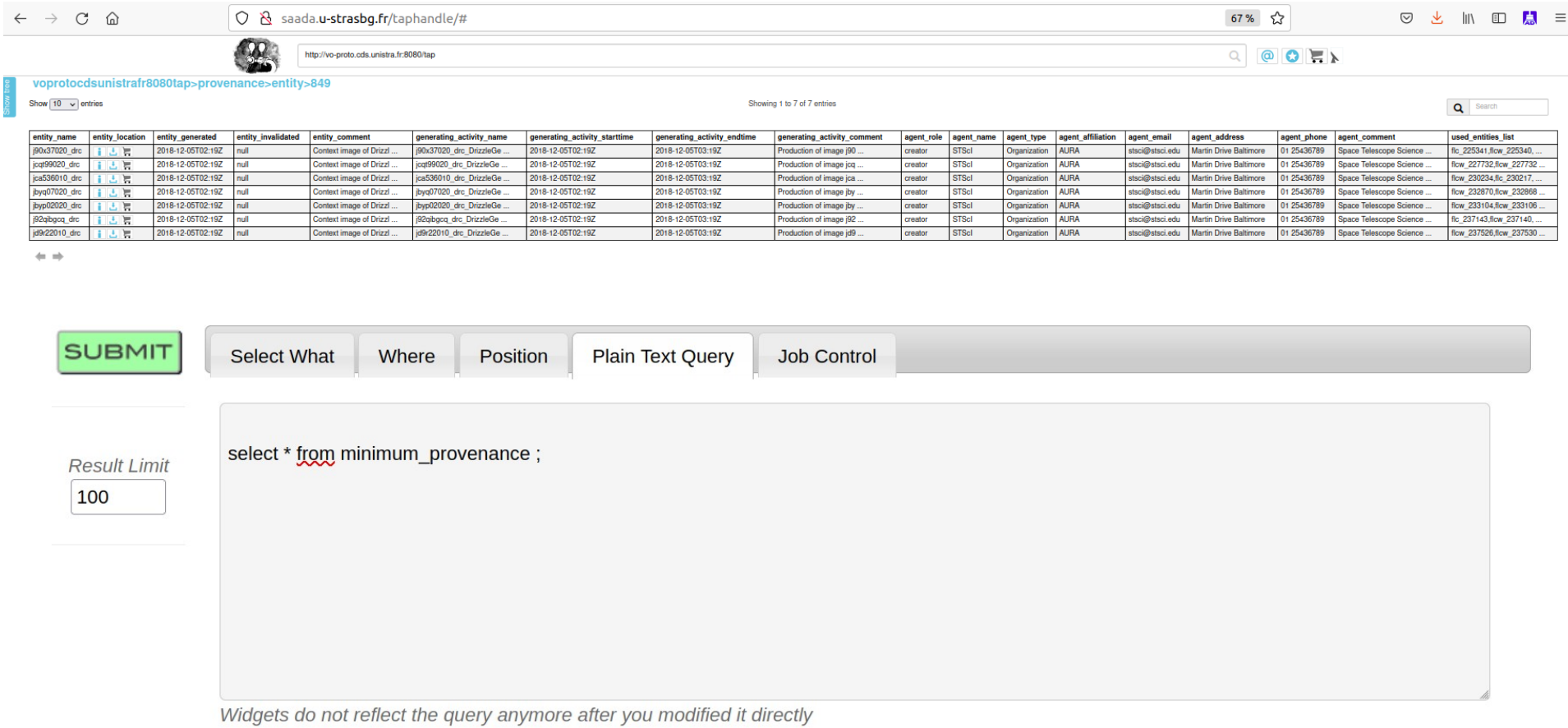
Single step = single table (= join)

- View (in postgres)

```
create view last_step_provenance as select

e.e_id AS entity_id,  e.e_name AS entity_name, e.e_location AS entity_location, e.e_generated AS entity_generated,
    e.e_invalidated AS entity_invalidated, e.e_comment AS entity_comment,

 activity.a_name AS generating_activity_name, activity.a_starttime AS generating_activity_starttime,
activity.a_endtime AS generating_activity_endtime, activity.a_comment AS generating_activity_comment,

 wasattributedto.wat_role AS agent_role, agent.ag_name AS agent_name, agent.ag_type AS agent_type,
    agent.ag_affiliation AS agent_affiliation, agent.ag_email AS agent_email, agent.ag_address AS agent_address,
    agent.ag_phone AS agent_phone, agent.ag_comment AS agent_comment,

string_agg(used.u_entity::text, ','::text) AS used_entities_list
FROM entity e

 JOIN wasgeneratedby ON e.e_id::text = wasgeneratedby.wgb_entity::text
 JOIN activity ON wasgeneratedby.wgb_activity::text = activity.a_id::text
join used on u_activity = a_id
join entity as ee on ee.e_id = u_entity
join wasattributedto on wat_entity = e.e_id
join agent on ag_id = wat_agent ;
```

Solutions

Single step = view query execution



Success and limitations

- Clear column names for distinct objects
- No more redundancy
- But :
  - Complex recursivity to manage
  - No direct retrieval for chains of provenance

Going Further

- TAP annotation of the simple table query
  (see Mireille's talk)
- Renormalized response (multitable)
- Instance query
  (see DaveMorris/Laurent Michel talk)
  → no simulation here

ProvTAP annotation

Denormalized solution