SimDB implementation (DEUVO) and SimDAL discussions

Observatoire de Paris / VO-Paris Data Centre

David Languignon, Benjamin Ooghe, Nicolas Moreau, Yann Rasera, Franck Le Petit, Jean-Michel Alimi

Napoli, Italy

May 16, 2011

Introduction

This presentation is organized as followed :

1. Description of the SimDB/DM implementation we made for the service "Deuvo"
2. Description of the successive steps we followed to try to build a DAL implementation to query the DM

This presentation is intended to :

1. Describe the experimentations we have made since last inter-op on SimDB DM and DAL
2. Present difficulties and problems we have encountered
3. Discuss solutions we have found to solve this problems

Table of contents

- Introduction
- SimDB DM Implementation
  - About DEUVO
  - DM Implementation
- SimDB DAL implementation, 1st try
  - Global Architecture
  - Interface specification
  - Problems and solutions
- SimDB DAL implementation, 2nd try
  - Creation of a first TAP access layer
    - Why and How ?
    - Orthogonalization principle
- SimDB implementation, next step
  - Proposals to solve current implementation problems
  - How could document oriented DB solve ingestion problems ?
  - How could document oriented DB solve query problem ?
  - To Do
- Conclusion

Introduction

SimDB DM Implementation

- About DEUVO
- DM Implementation

SimDB DAL implementation, 1st try

SimDB DAL implementation, 2nd try

SimDB implementation, next step

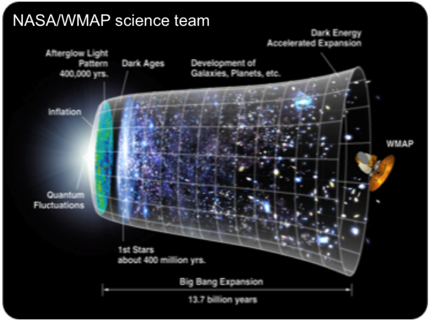
Conclusion

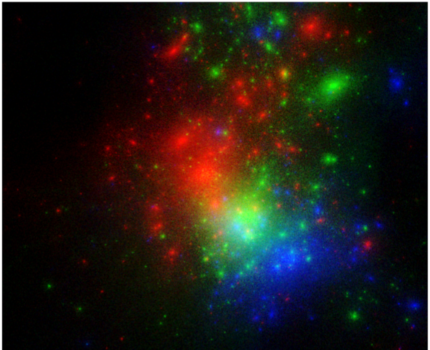
About DEUVO

DEUVO : "Dark Energy Universe Virtual Observatory"

The project aims at **investigating** the imprints of dark energy on cosmic structure formation **through very high resolution cosmological simulations**

- 9 simulations with 1 billion particles
- 5 000 000 CPU hours (600 years)
- 40 Tb data produced
- Post-processing : 500 000 halos / simulation


NASA/WMAP science team
Dark Energy Accelerated Expansion
Afterglow Light Pattern 400,000 yrs.
Dark Ages
Development of Galaxies, Planets, etc.
Inflation
WMAP
Quantum Fluctuations
1st Stars about 400 million yrs.
Big Bang Expansion
13.7 billion years



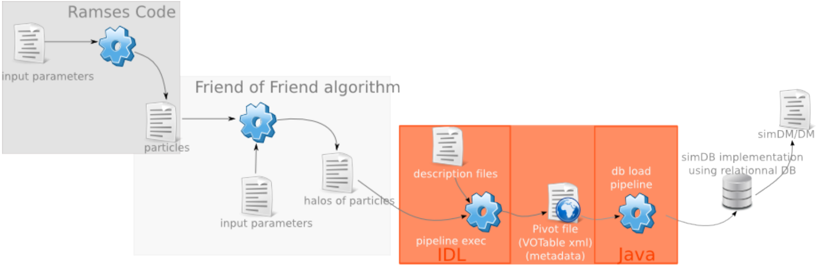
DM Implementation

Based on VO-Paris past experiences (StarFormat and PDR implementation)

- based on vo-urp
- custom ingestion pipeline for (meta-)data loading
  - 1 postprocessing produce more than 100 000 products (halos)
  - Very high RAM/CPU requirements to load (meta-)data into DB

DM Implementation

Ingestion pipeline



Introduction

SimDB DM Implementation

SimDB DAL implementation, 1st try

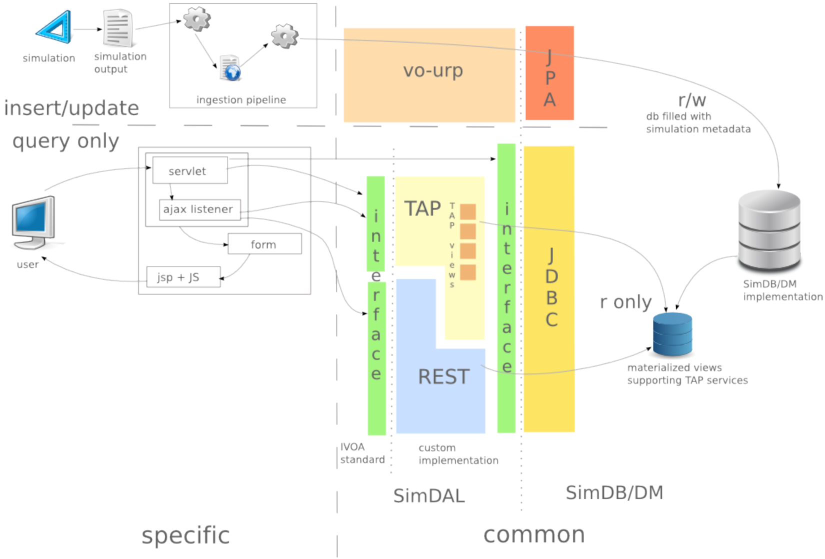
- Global Architecture
- Interface specification
- Problems and solutions

SimDB DAL implementation, 2nd try

SimDB implementation, next step

Conclusion

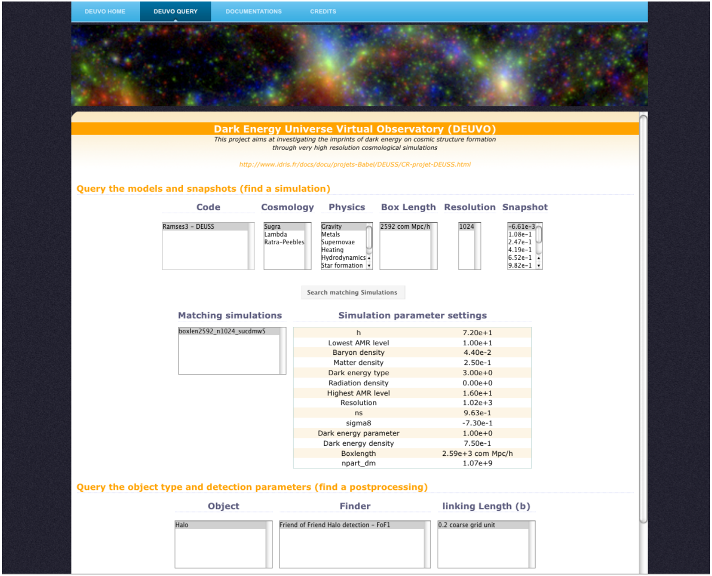
Global Architecture



Interface specification

- Common queries using JDBC implementation (cf R.Wagner SimDAP operations proposals, 2009 Strasbourg)
  - getProjectList()
  - getProtocolList() (→ listProtocols)
  - getInputParameters(protocolId)
  - ...
- rawQuery() function (→ queryData)

Deuvo UI simulation search



Dark Energy Universe Virtual Observatory (DEUVO)

This project aims at investigating the imprints of dark energy on cosmic structure formation through very high resolution cosmological simulations

http://www.idris.fr/docs/docu/projets-Babel/DEUSS/CR-projet-DEUSS.html

Query the models and snapshots (find a simulation)

Code	Cosmology	Physics	Box Length	Resolution	Snapshot
Ramses3 - DEUSS	Sugra Lambda Ratra-Peebles	Gravity Metals Supernovae Heating Hydrodynamics Star formation	2592 com Mpc/h	1024	-6.61e-3 1.08e-1 2.47e-1 4.19e-1 6.52e-1 9.82e-1

Matching simulations

boxlen2592_n1024_sucdmw5

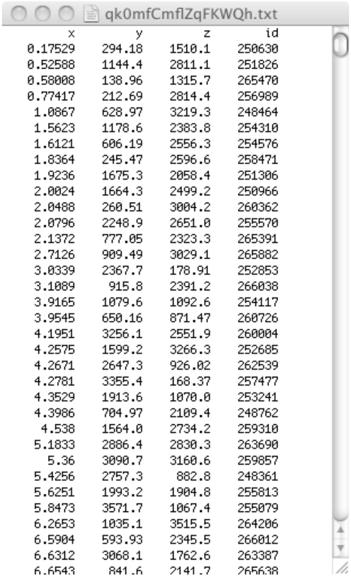
Simulation parameter settings

h	7.20e+1
Lowest AMR level	1.00e+1
Baryon density	4.40e-2
Matter density	2.50e-1
Dark energy type	3.00e+0
Radiation density	0.00e+0
Highest AMR level	1.60e+1
Resolution	1.02e+3
ns	9.63e-1
sigma8	-7.30e-1
Dark energy parameter	1.00e+0
Dark energy density	7.50e-1
Boxlength	2.59e+3 com Mpc/h
npart_dm	1.07e+9

Query the object type and detection parameters (find a postprocessing)

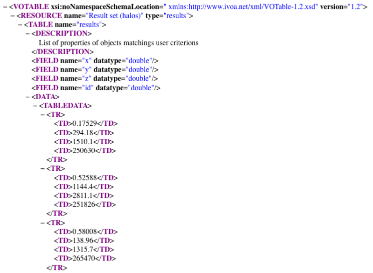
Object	Finder	linking Length (b)
Halo	Friend of Friend Halo detection - FoF1	0.2 coarse grid unit

Deuvo UI output formats



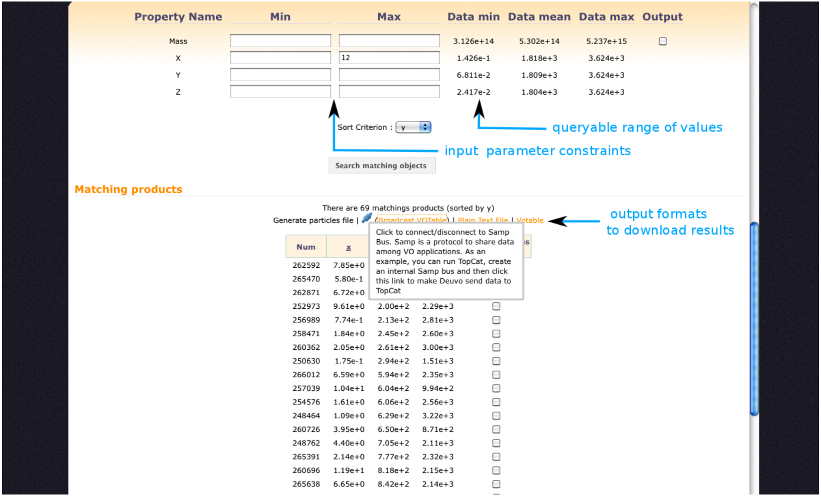
qk0mfCmflZqFKWQh.txt

```
       x        y        z       id
 0.17529   294.18   1510.1   250630
 0.52588   1144.4   2811.1   251826
 0.58008   138.96   1315.7   265470
 0.77417   212.69   2814.4   256989
  1.0867   628.97   3219.3   248464
  1.5623   1178.6   2383.8   254310
  1.6121   606.19   2556.3   254576
  1.8364   245.47   2596.6   258471
  1.9236   1675.3   2058.4   251306
  2.0024   1664.3   2499.2   250966
  2.0488   260.51   3004.2   260362
  2.0796   2248.9   2651.0   255570
  2.1372   777.05   2323.3   265391
  2.7126   909.49   3029.1   265882
  3.0339   2367.7   178.91   252853
  3.1089    915.8   2391.2   266038
  3.9165   1079.6   1092.6   254117
  3.9545   650.16   871.47   260726
  4.1951   3256.1   2551.9   260004
  4.2575   1599.2   3266.3   252685
  4.2671   2647.3   926.02   262539
  4.2781   3355.4   168.37   257477
  4.3529   1913.6   1070.0   253241
  4.3986   704.97   2109.4   248762
   4.538   1564.0   2734.2   259310
  5.1833   2886.4   2830.3   263690
    5.36   3090.7   3160.6   259857
  5.4256   2757.3    882.8   248361
  5.6251   1993.2   1904.8   255813
  5.8473   3571.7   1067.4   255079
  6.2653   1035.1   3515.5   264206
  6.5904   593.93   2345.5   266012
  6.6312   3068.1   1762.6   263387
  6.6543    841.6   2141.7   265638
```



```
– <VOTABLE xsi:noNamespaceSchemaLocation=" xmlns:http://www.ivoa.net/xml/VOTable-1.2.xsd" version="1.2">
  – <RESOURCE name="Result set (halos)" type="results">
    – <TABLE name="results">
      – <DESCRIPTION>
          List of properties of objects matchings user criterions
        </DESCRIPTION>
        <FIELD name="x" datatype="double"/>
        <FIELD name="y" datatype="double"/>
        <FIELD name="z" datatype="double"/>
        <FIELD name="id" datatype="double"/>
      – <DATA>
        – <TABLEDATA>
          – <TR>
              <TD>0.17529</TD>
              <TD>294.18</TD>
              <TD>1510.1</TD>
              <TD>250630</TD>
            </TR>
          – <TR>
              <TD>0.52588</TD>
              <TD>1144.4</TD>
              <TD>2811.1</TD>
              <TD>251826</TD>
            </TR>
          – <TR>
              <TD>0.58008</TD>
              <TD>138.96</TD>
              <TD>1315.7</TD>
              <TD>265470</TD>
            </TR>
```

Deuvo UI product search with contraints



Property Name	Min	Max	Data min	Data mean	Data max	Output
Mass			3.126e+14	5.302e+14	5.237e+15	☐
X		12	1.426e-1	1.818e+3	3.624e+3	
Y			6.811e-2	1.809e+3	3.624e+3	
Z			2.417e-2	1.804e+3	3.624e+3	

Sort Criterion : y

Search matching objects

queryable range of values

input parameter constraints

Matching products

There are 69 matchings products (sorted by y)

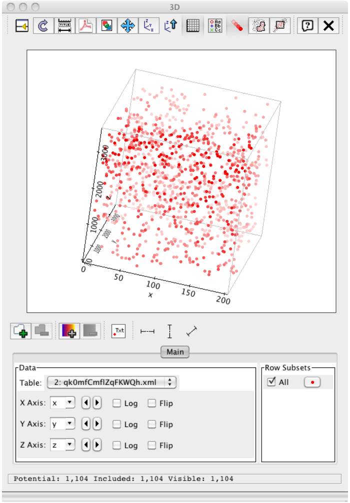
Generate particles file | (Broadcast VOTable) | Plain Text File | Votable

output formats to download results

Click to connect/disconnect to Samp Bus. Samp is a protocol to share data among VO applications. As an example, you can run TopCat, create an internal Samp bus and then click this link to make Deuvo send data to TopCat

Num	x			s
262592	7.85e+0			
265470	5.80e-1			
262871	6.72e+0			
252973	9.61e+0	2.00e+2	2.29e+3	☐
256989	7.74e-1	2.13e+2	2.81e+3	☐
258471	1.84e+0	2.45e+2	2.60e+3	☐
260362	2.05e+0	2.61e+2	3.00e+3	☐
250630	1.75e-1	2.94e+2	1.51e+3	☐
266012	6.59e+0	5.94e+2	2.35e+3	☐
257039	1.04e+1	6.04e+2	9.94e+2	☐
254576	1.61e+0	6.06e+2	2.56e+3	☐
248464	1.09e+0	6.29e+2	3.22e+3	☐
260726	3.95e+0	6.50e+2	8.71e+2	☐
248762	4.40e+0	7.05e+2	2.11e+3	☐
265391	2.14e+0	7.77e+2	2.32e+3	☐
260696	1.19e+1	8.18e+2	2.15e+3	☐
265638	6.65e+0	8.42e+2	2.14e+3	☐

TopCat visualisation from Deuvo data through Samp protocol



Data

Table: 2: qk0mfCmflZqFKWQh.xml

X Axis: x □ Log □ Flip

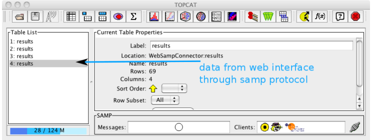
Y Axis: y □ Log □ Flip

Z Axis: z □ Log □ Flip

Row Subsets

☑ All

```
Potential: 1,104 Included: 1,104 Visible: 1,104
```



1: results
2: results
3: results
4: results

Label: results
Location: WebSampConnector:results
Name: results
Rows: 69
Columns: 4
Sort Order:
Row Subset: All

Messages:
Clients:

data from web interface through samp protocol

Problems and solutions

Problems :

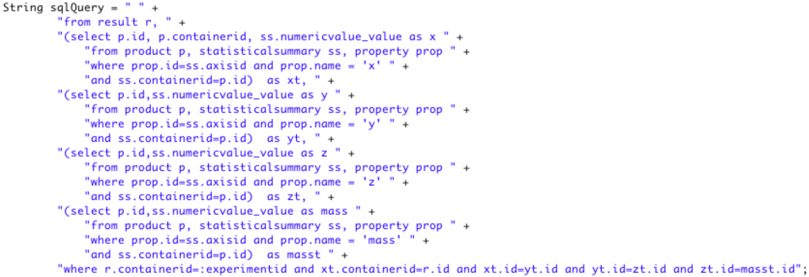
- SimDB DataModel implies a lot of **high cost JOIN (in terms of CPU/time)** when implemented using a relational design (ex : product × statisticalSummary)
- Basic interface (too much), with very few predefined queries and not IVOA Standard compliant (neither REst nor TAP)

Solutions :

- Rewrite SQL queries to fit postgresql optimizer choices
- Tune postgresql (query optimizer, memory management, disk i/o)
- Define proper indexes
- Have queries go through protocol side of the DM instead of experiment one

Example of products query matching only 4 constraints...

Each constraint addition implies an heavy JOIN (x,y,z,mass)



```
String sqlQuery = " " +
        "from result r, " +
        "(select p.id, p.containerid, ss.numericvalue_value as x " +
            "from product p, statisticalsummary ss, property prop " +
            "where prop.id=ss.axisid and prop.name = 'x' " +
            "and ss.containerid=p.id)  as xt, " +
        "(select p.id,ss.numericvalue_value as y " +
            "from product p, statisticalsummary ss, property prop " +
            "where prop.id=ss.axisid and prop.name = 'y' " +
            "and ss.containerid=p.id)  as yt, " +
        "(select p.id,ss.numericvalue_value as z " +
            "from product p, statisticalsummary ss, property prop " +
            "where prop.id=ss.axisid and prop.name = 'z' " +
            "and ss.containerid=p.id)  as zt, " +
        "(select p.id,ss.numericvalue_value as mass " +
            "from product p, statisticalsummary ss, property prop " +
            "where prop.id=ss.axisid and prop.name = 'mass' " +
            "and ss.containerid=p.id)  as masst " +
        "where r.containerid=:experimentid and xt.containerid=r.id and xt.id=yt.id and yt.id=zt.id and zt.id=masst.id";
```

Introduction

SimDB DM Implementation

SimDB DAL implementation, 1^{st} try

SimDB DAL implementation, 2^{nd} try

- Creation of a first TAP access layer
  - Why and How ?
  - Orthogonalization principle

SimDB implementation, next step

Conclusion

Creation of a first TAP access layer 1/3

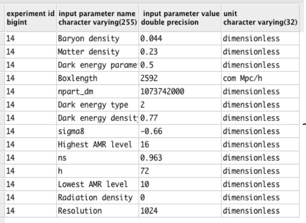
Why and How ?

- To make SimDB relationnal database implementation queryable :
  - in terms of performance
  - in terms of ease of query writing
- Through an implementation of a subset of the TheoryGroup proposal at Nara (G.Lemson)
  - Table orthogonalization using automated procedure (script)
    - **product** × statisticalsummary
    - **protocol** × inputparameter

Creation of a first TAP access layer 2/3

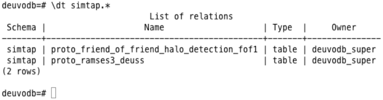
Orthogonalization principle

standard simDB configuration



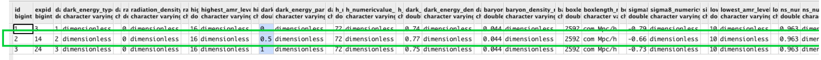
experiment id bigint	input parameter name character varying(255)	input parameter value double precision	unit character varying(32)
14	Baryon density	0.044	dimensionless
14	Matter density	0.23	dimensionless
14	Dark energy parame	0.5	dimensionless
14	Boxlength	2592	com Mpc/h
14	npart_dm	1073742000	dimensionless
14	Dark energy type	2	dimensionless
14	Dark energy densit	0.77	dimensionless
14	sigma8	-0.66	dimensionless
14	Highest AMR level	16	dimensionless
14	ns	0.963	dimensionless
14	h	72	dimensionless
14	Lowest AMR level	10	dimensionless
14	Radiation density	0	dimensionless
14	Resolution	1024	dimensionless

one table per protocol



```
deuvodb=# \dt simtap.*
                              List of relations
 Schema |                   Name                    | Type  |     Owner
--------+-------------------------------------------+-------+--------------
 simtap | proto_friend_of_friend_halo_detection_fof1 | table | deuvodb_super
 simtap | proto_ramses3_deuss                        | table | deuvodb_super
(2 rows)

deuvodb=# 
```



id	expid	da	dark_energy_typ	da	ra	radiation_density	ra	hig	highest_amr_leve	hi	dark	dark_energy_par	da	h_	h_numericvalue_	h_	dark_	dark_energy_den	da	baryon	baryon_density_	ba	boxle	boxlength_r	bo	sigma	sigma8_numeric	si	low	lowest_amr_level	lo	ns_nur	ns_nu
bigint	bigint	dc	character varying	ch	dc	character varying	ch	do	character varying	ch	doul	character varying	ch	do	character varying	ch	doub	character varying	ch	double	character varying	ch	doub	character va	ch	double	character varying	ch	do	character varying	ch	double	chara
1	3	1	dimensionless		0	dimensionless		16	dimensionless		0	dimensionless		72	dimensionless		0.74	dimensionless		0.044	dimensionless		2592	com Mpc/h		-0.79	dimensionless		10	dimensionless		0.963	dimer
2	14	2	dimensionless		0	dimensionless		16	dimensionless		0.5	dimensionless		72	dimensionless		0.77	dimensionless		0.044	dimensionless		2592	com Mpc/h		-0.66	dimensionless		10	dimensionless		0.963	dimer
3	24	3	dimensionless		0	dimensionless		16	dimensionless		1	dimensionless		72	dimensionless		0.75	dimensionless		0.044	dimensionless		2592	com Mpc/h		-0.73	dimensionless		10	dimensionless		0.963	dimer

orthogonalized SimTAP configuration

Creation of a first TAP access layer 3/3

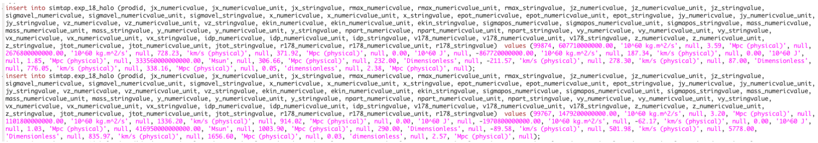
Problems

- The solution is not scalable
  - as many columns as the number of inputparameter or statisticalsummary associated with product/protocol
    - Can be $\ggg$ 100 !!
    - Impossible to describe queryable columns list to user
    - Impossible to write queries against such a table with reasonable ease
- Need custom/specific TAP table definition for each particular use of SimDB
  - UI dependant, not SimDB dependant

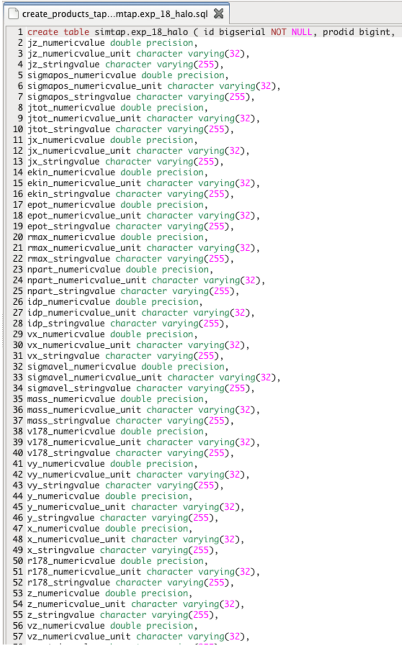
It appears

that SimDB/DM implementation using a relationnal DB is not efficient to provide reasonably fast user-friendly access to simulation (meta)data

Output of build_tap_products_views.py

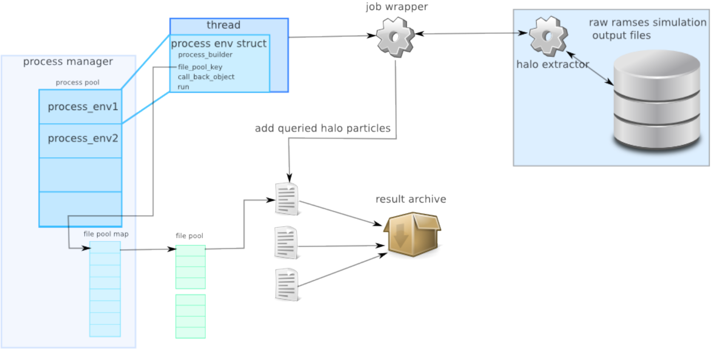


```
insert into simtap.exp_18_halo (prodid, jx_numericvalue, jx_numericvalue_unit, jx_stringvalue, rmax_numericvalue, rmax_numericvalue_unit, rmax_stringvalue, jz_numericvalue, jz_numericvalue_unit, jz_stringvalue,
sigmavel_numericvalue, sigmavel_numericvalue_unit, sigmavel_stringvalue, x_numericvalue, x_numericvalue_unit, x_stringvalue, epot_numericvalue, epot_numericvalue_unit, epot_stringvalue, jy_numericvalue, jy_numericvalue_unit,
jy_stringvalue, vz_numericvalue, vz_numericvalue_unit, vz_stringvalue, ekin_numericvalue, ekin_numericvalue_unit, ekin_stringvalue, sigmapos_numericvalue, sigmapos_numericvalue_unit, sigmapos_stringvalue, mass_numericvalue,
mass_numericvalue_unit, mass_stringvalue, y_numericvalue, y_numericvalue_unit, y_stringvalue, npart_numericvalue, npart_numericvalue_unit, npart_stringvalue, vy_numericvalue, vy_numericvalue_unit, vy_stringvalue,
vx_numericvalue, vx_numericvalue_unit, vx_stringvalue, idp_numericvalue, idp_numericvalue_unit, idp_stringvalue, v178_numericvalue, v178_numericvalue_unit, v178_stringvalue, z_numericvalue, z_numericvalue_unit,
z_stringvalue, jtot_numericvalue, jtot_numericvalue_unit, jtot_stringvalue, r178_numericvalue, r178_numericvalue_unit, r178_stringvalue)  values (99874, 607710000000.00, '10^60 kg.m^2/s', null, 3.59, 'Mpc (physical)', null,
2676800000000.00, '10^60 kg.m^2/s', null, 728.23, 'km/s (physical)', null, 371.92, 'Mpc (physical)', null, 0.00, '10^60 J', null, -867720000000.00, '10^60 kg.m^2/s', null, 187.34, 'km/s (physical)', null, 0.00, '10^60 J',
null, 1.85, 'Mpc (physical)', null, 333560000000000.00, 'Msun', null, 306.66, 'Mpc (physical)', null, 232.00, 'Dimensionless', null, -211.57, 'km/s (physical)', null, 278.30, 'km/s (physical)', null, 87.00, 'Dimensionless',
null, 776.05, 'km/s (physical)', null, 338.16, 'Mpc (physical)', null, 0.05, 'dimensionless', null, 2.38, 'Mpc (physical)', null);
insert into simtap.exp_18_halo (prodid, jx_numericvalue, jx_numericvalue_unit, jx_stringvalue, rmax_numericvalue, rmax_numericvalue_unit, rmax_stringvalue, jz_numericvalue, jz_numericvalue_unit, jz_stringvalue,
sigmavel_numericvalue, sigmavel_numericvalue_unit, sigmavel_stringvalue, x_numericvalue, x_numericvalue_unit, x_stringvalue, epot_numericvalue, epot_numericvalue_unit, epot_stringvalue, jy_numericvalue, jy_numericvalue_unit,
jy_stringvalue, vz_numericvalue, vz_numericvalue_unit, vz_stringvalue, ekin_numericvalue, ekin_numericvalue_unit, ekin_stringvalue, sigmapos_numericvalue, sigmapos_numericvalue_unit, sigmapos_stringvalue, mass_numericvalue,
mass_numericvalue_unit, mass_stringvalue, y_numericvalue, y_numericvalue_unit, y_stringvalue, npart_numericvalue, npart_numericvalue_unit, npart_stringvalue, vy_numericvalue, vy_numericvalue_unit, vy_stringvalue,
vx_numericvalue, vx_numericvalue_unit, vx_stringvalue, idp_numericvalue, idp_numericvalue_unit, idp_stringvalue, v178_numericvalue, v178_numericvalue_unit, v178_stringvalue, z_numericvalue, z_numericvalue_unit,
z_stringvalue, jtot_numericvalue, jtot_numericvalue_unit, jtot_stringvalue, r178_numericvalue, r178_numericvalue_unit, r178_stringvalue)  values (99767, 1479200000000.00, '10^60 kg.m^2/s', null, 3.20, 'Mpc (physical)', null,
1101800000000.00, '10^60 kg.m^2/s', null, 1336.20, 'km/s (physical)', null, 914.02, 'Mpc (physical)', null, 0.00, '10^60 J', null, -1970800000000.00, '10^60 kg.m^2/s', null, -62.17, 'km/s (physical)', null, 0.00, '10^60 J',
null, 1.03, 'Mpc (physical)', null, 416950000000000.00, 'Msun', null, 1003.90, 'Mpc (physical)', null, 290.00, 'Dimensionless', null, -89.58, 'km/s (physical)', null, 501.98, 'km/s (physical)', null, 5778.00,
'Dimensionless', null, 835.97, 'km/s (physical)', null, 1656.60, 'Mpc (physical)', null, 0.03, 'dimensionless', null, 2.57, 'Mpc (physical)', null);
```



```sql
create table simtap.exp_18_halo ( id bigserial NOT NULL, prodid bigint,
jz_numericvalue double precision,
jz_numericvalue_unit character varying(32),
jz_stringvalue character varying(255),
sigmapos_numericvalue double precision,
sigmapos_numericvalue_unit character varying(32),
sigmapos_stringvalue character varying(255),
jtot_numericvalue double precision,
jtot_numericvalue_unit character varying(32),
jtot_stringvalue character varying(255),
jx_numericvalue double precision,
jx_numericvalue_unit character varying(32),
jx_stringvalue character varying(255),
ekin_numericvalue double precision,
ekin_numericvalue_unit character varying(32),
ekin_stringvalue character varying(255),
epot_numericvalue double precision,
epot_numericvalue_unit character varying(32),
epot_stringvalue character varying(255),
rmax_numericvalue double precision,
rmax_numericvalue_unit character varying(32),
rmax_stringvalue character varying(255),
npart_numericvalue double precision,
npart_numericvalue_unit character varying(32),
npart_stringvalue character varying(255),
idp_numericvalue double precision,
idp_numericvalue_unit character varying(32),
idp_stringvalue character varying(255),
vx_numericvalue double precision,
vx_numericvalue_unit character varying(32),
vx_stringvalue character varying(255),
sigmavel_numericvalue double precision,
sigmavel_numericvalue_unit character varying(32),
sigmavel_stringvalue character varying(255),
mass_numericvalue double precision,
mass_numericvalue_unit character varying(32),
mass_stringvalue character varying(255),
v178_numericvalue double precision,
v178_numericvalue_unit character varying(32),
v178_stringvalue character varying(255),
vy_numericvalue double precision,
vy_numericvalue_unit character varying(32),
vy_stringvalue character varying(255),
y_numericvalue double precision,
y_numericvalue_unit character varying(32),
y_stringvalue character varying(255),
x_numericvalue double precision,
x_numericvalue_unit character varying(32),
x_stringvalue character varying(255),
r178_numericvalue double precision,
r178_numericvalue_unit character varying(32),
r178_stringvalue character varying(255),
z_numericvalue double precision,
z_numericvalue_unit character varying(32),
z_stringvalue character varying(255),
vz_numericvalue double precision,
vz_numericvalue_unit character varying(32),
```

Prototype of a raw data access RESTful webservice



Introduction

SimDB DM Implementation

SimDB DAL implementation, 1^{st} try

SimDB DAL implementation, 2^{nd} try

SimDB implementation, next step

- Proposals to solve current implementation problems
- How could document oriented DB solve ingestion problems ?
- How could document oriented DB solve query problem ?
- To Do

Conclusion

Proposals to solve current implementation problems

R.Wagner, Strasbourg 2009 :

- Queries for existing data are really queries against the data model, not its implementation.
- Using the data model doesnt require managing a relational database.

Our own experience :

- DM is efficient to describe data in an object way (nested structures → UML compositions).
- Tries to fit the object oriented DM in tabular structures :
  - makes load and query job difficult
  - require (too much) successive data transformations

It appears that

Document oriented DB (or object oriented), closer to the DM design, could be the solution

Think about document oriented DB I

Pro :

- no need for TAP or materialized views against the main SimDB/DM implementation
- Easy and consistent data organisation (make query easy)
- Solve composition problem by removing referential integrity needs
- Easily usable with many languages
- Some solutions use BSON (and so are natively JSON-capable)
  - JSON could be used through a new ingestion pipeline directly from simulation (JSON output or XML+XSLT) to JSON Document oriented SimDB/DM implementation

Think about document oriented DB II

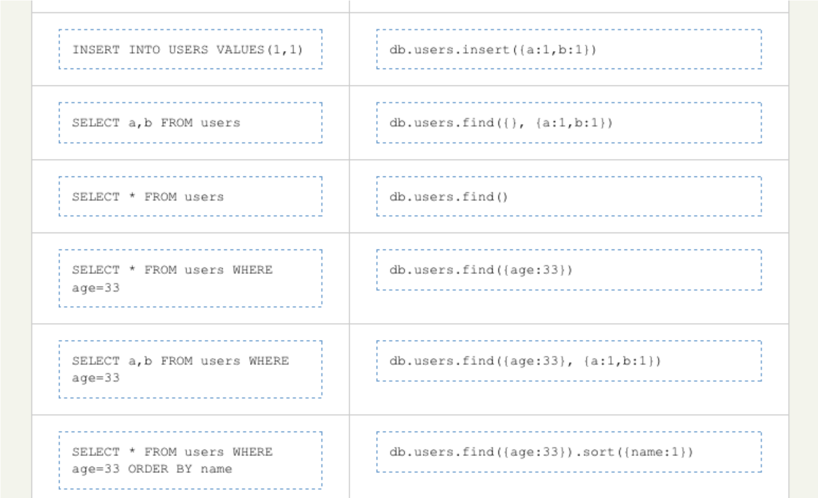
- Easily scalable (allow MapReduce, GridFS use)
- comply with Rick Wagner recommendation in Strasbourg IVOA inter-op : to follow a nested elements way

Cons :

- Not a relationnal tabular structure ⟶ ease of a SimTAP implementation (it's noSQL, so noADQL...) ?
- No validation layer in standard for JSON format

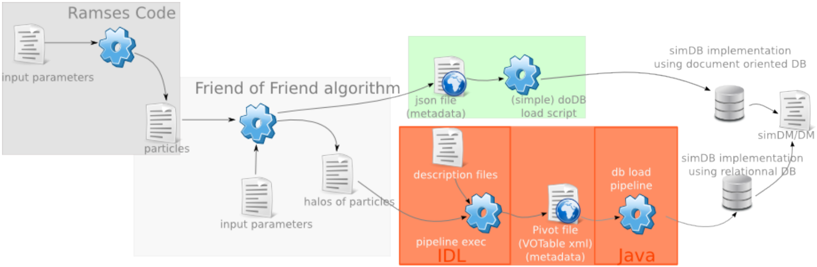
Example of dodb ⟷ tabular query mapping

From the mongodb project

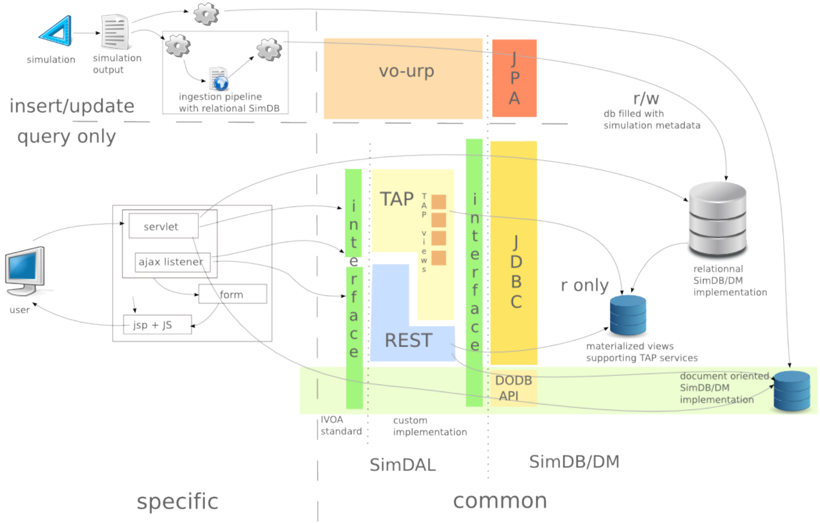


`INSERT INTO USERS VALUES(1,1)`	`db.users.insert({a:1,b:1})`
`SELECT a,b FROM users`	`db.users.find({}, {a:1,b:1})`
`SELECT * FROM users`	`db.users.find()`
`SELECT * FROM users WHERE age=33`	`db.users.find({age:33})`
`SELECT a,b FROM users WHERE age=33`	`db.users.find({age:33}, {a:1,b:1})`
`SELECT * FROM users WHERE age=33 ORDER BY name`	`db.users.find({age:33}).sort({name:1})`

How could document oriented DB solve ingestion problem ?



How could document oriented DB solve query problem ?



To Do

- This solution needs a prototype implementation to evaluate :
    - Queries performance
    - Usability (in terms of ease of query writing, ability to fit IVOA requirements)

Conclusion

- DM is robust, abstract enough to deal with all the simulations we adressed
- The first DM implementation (led by vo-urp) is effective to turn DM into technical reality but not to allow realistic interactive query job.
- Implementations of the DM using a relationnal/tabular design have shown some limitations
  - Too many high cpu cost JOIN between tables (for queries but also data loading)
  - Difficulties to implement a DAL layer based on tables (cf orthogonalization tests)
  - Scalability

Conclusion

- Today we can take two different directions
  - Continue with the relational way, in this case we have to deal with the JOIN problem which seems to be difficult to solve since inherent to tabular design.
  - Prototype a new implementation using a design closer to the object oriented nature of the SimDB/DM (job currently being done at Meudon Observatory).