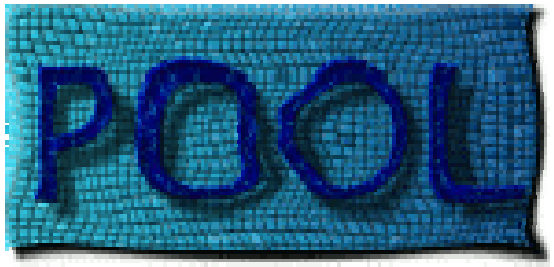


Object Storage into RDBMS through the POOL framework

Ioannis Papadopoulos, CERN IT/ADC

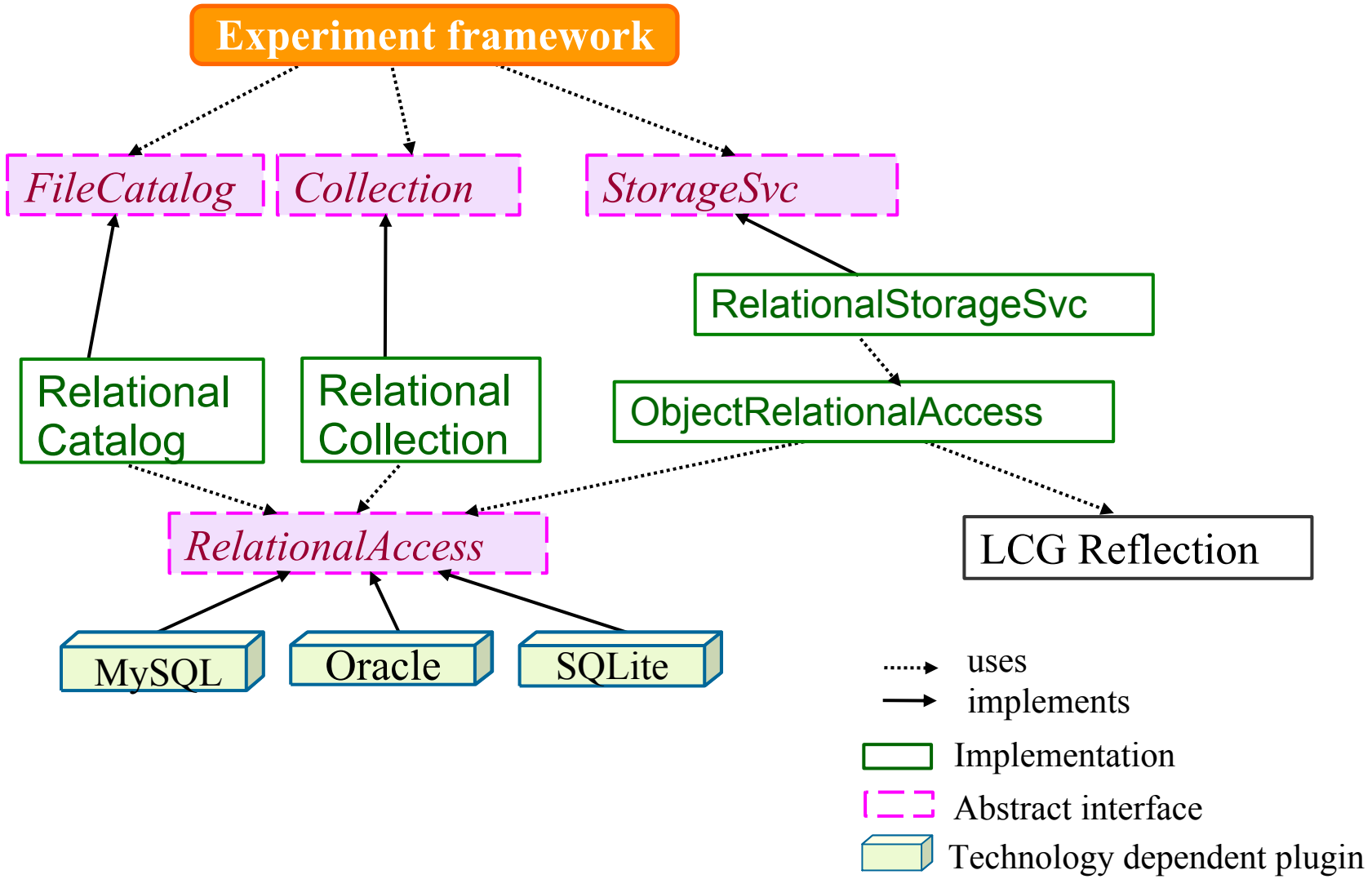
http://pool.cern.ch

January 27th, 2005

Motivation for RDBMS-based object I/O

- On-line configuration/conditions data
  - Present data already written into an RDBMS as C++ objects in the off-line reconstruction/analysis framework
- Off-line conditions/calibration/geometry data
  - Same technology for IoV index and data payload

Reminder: RAL in POOL



Object Storage using RAL and POOL

- ObjectRelationalAccess
  - Bridging the differences between object and relational worlds
  - Connection with the SEAL dictionary
- RelationalStorageService
  - Implementation of the POOL StorageSvc developer-level interfaces based on the ObjectRelationalAccess package
- Command line tools
  - To accommodate existing schemas and relational data
  - To customize the object view of the relational data (and inversely)

Object storage into RDBMS

- How to map classes ↔ tables ?
  - Both C++ and SQL allow the description of data layout
  - …but with very different constraints/aims
    - no single unique mapping
    - need to store object/relational mapping together with object data
- No notion of object identity in RDBMS (persistent address)
  - requires unique index for addressable objects
  - part of mapping definition

A Mapping Example (I)


```
class A {
   int x;
   float y;
   std::vector<double> v;
   class B {
      int i;
      std::string s;
   } b;
};
```

A Mapping Example (II)

p.k. *f.k. constraint*

T_A

ID	X	Y	B_I	B_S
1	10	1.4	3	“Hello”
2	22	2.2	3	“Hi”
.	.	.	.	.

T_A_V

ID	POS	V
1	1	0.12
1	2	12.2
1	3	4.1
1	4	5.452
2	1	32.1
2	2	0.1
2	3	0.1

This is only one of the possible mappings!

defining/storing/materializing mappings

- The ObjectRelationalAccess package provides
  - Definition of hierarchy of transient mapping elements
    - Element types: Object, Primitive, Array, Pointer, PoolReference
    - Element : variable type, name, scope, columns, and sub-elements
  - Default object/relational mapping generation
    - Takes care of duplication and lengths of table/column names
  - Persistency of mapping definition
    - Versioning
    - Transient structure stored in three tables
  - Materialization of mapping (schema preparation)
    - Ensures the generation of proper indices and constraints
- Command line tools
  - Customized mapping using an XML driver file
  - Dumping the mapping information into XML files

The RelationalStorageService

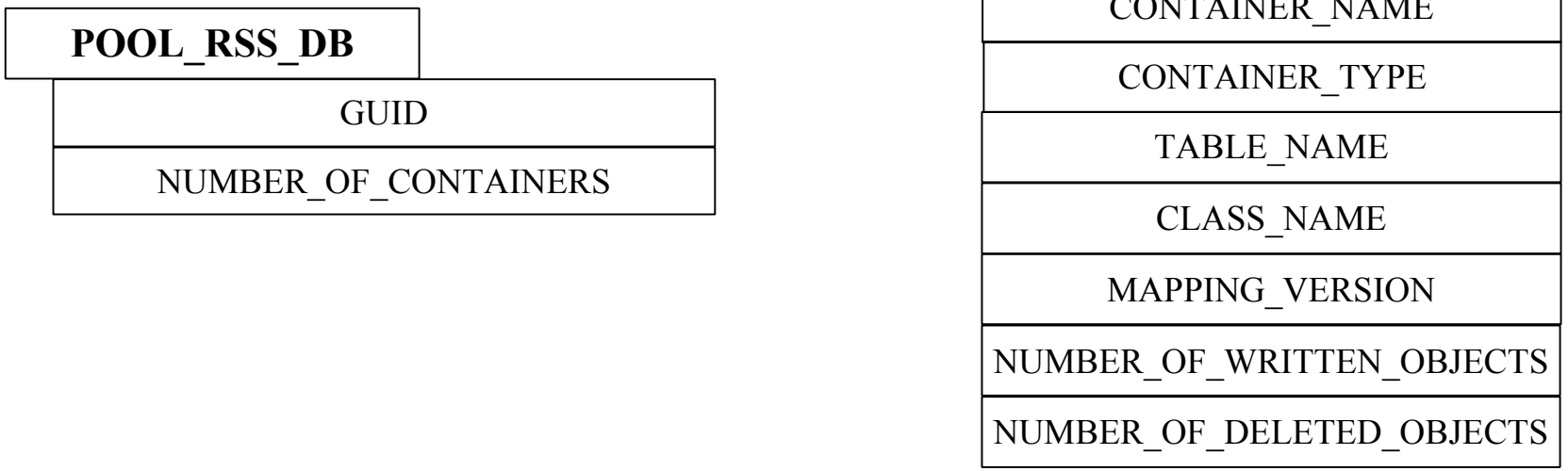
- Designed to make use of the full functionality of the POOL StorageSvc framework
  - Reminder : POOL data hierarchy:
    - Technology domain
    - Database (ROOT or SQLite file, MySQL database, Oracle schema)
    - Container (collection of physically or logically clustered objects)
    - Object ID
- Appears in latest internal releases of POOL
- Supports two minor technologies
  - POOL_RDBMS_HOMOGENEOUS (equivalent to ROOTTREE)
  - POOL_RDBMS_POLYMORPHIC (equivalent to ROOTKEY)

Current capabilities

- Can store objects containing:
  - embedded objects
  - STL containers (nested containment as well)
  - pool::Reference types
- Not yet supported:
  - C-arrays
  - pointers
  - bitsets
  - long long (RAL limitation)
- Plugins
  - Oracle : fully functional
  - MySQL, SQLite : functional for objects with up to single level of STL containment

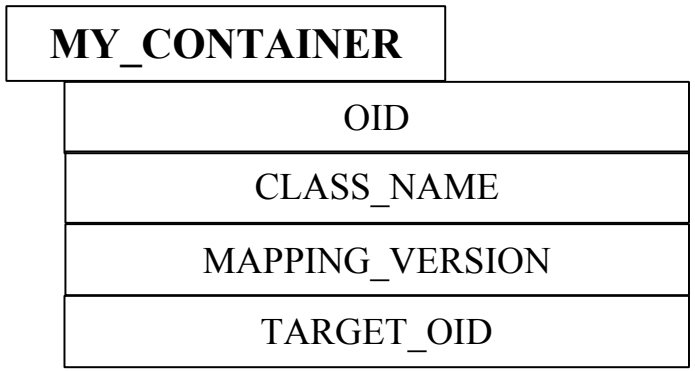
Implementation (I)

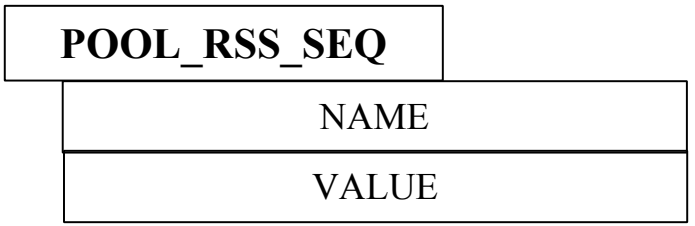
- Automatic generation of object/relational mappings
- Protection from concurrent writing through row locking
  - Locking of the container header table rows
  - Locking of the database table row



Implementation (II)

- Consistent reading guaranteed using read-only transactions
- Use of bind variables everywhere
- Early preparation and reuse of data buffers
- Homogeneous containers
  - Container table = Top-level table from class mapping
- Polymorphic containers
  - Container table = Table of object headers
  - Target OIDs controlled by “hand-made” sequences





Implementation (III)

- STL container I/O
  - Makes use of the SEAL dictionary information
    - Writing based on the existence of the “begin” and “size” methods of the container, and the “*” and “++” operators of the corresponding iterator
    - Reading based on the existence of the “insert(elem,pos)” method in every container
    - Special containers (queue, stack) handled through the corresponding underlying containers
    - Dictionary generation requires
      - “—pool” flag switched off
      - declaration of the iterator (and pair in case of maps) classes in the xml file
      - forward declaration of the iterator types
  - Bulk inserts for “leaf” containers when writing
  - Row pre-fetching when reading

The TODO list

- Command line tools for populating POOL containers from existing relational data
- Implement general selections (like in ROOTTREE)
- Enable streaming of large arrays into BLOBS
- Handle object identities made of more than a single column :
  - Registered plugins performing transformations from multiple values into a unique unsigned long type

Summary

- Object storage into RDBMS using the POOL storage mechanism has been achieved using the RAL
- Technologically neutral object description in RDBMS is feasible
- The RelationalStorageService is by itself a stress test case for the RAL plugins