The POOL relational abstraction layer and the relational storage manager

Ioannis Papadopoulos, for the POOL Team

LCG Applications Area Meeting 28/7/2004

Why a relational abstraction layer

- To achieve technology neutrality for the implementation of the relational components of POOL (and eventually the ConditionsDB)
  - less maintenance load
  - easy introductions/evaluations of new RDBMS flavours as backends for the POOL components
- To address the problem of distributing data in RDBMS of different flavours

Why a relational back end for the object storage

- More natural choice for non-event data
  - conditions, calibration/alignment
- To allow the "viewing" of existing relational data as C++ objects in the off-line reconstruction/analysis framework
  - a typical case: conditions data taken on-line.

The software project (I)

- Requirements collection and component analysis
  - Late 2003 till end of March 2004
  - Input from experts from CMS and ATLAS (Martin, Ad, Vincenzo, Torre, ...)
  - Requirements captured as use cases
  - Drafted a requirements and analysis document together with the clients
    - Domain decomposition
    - Derivation of main software requirements

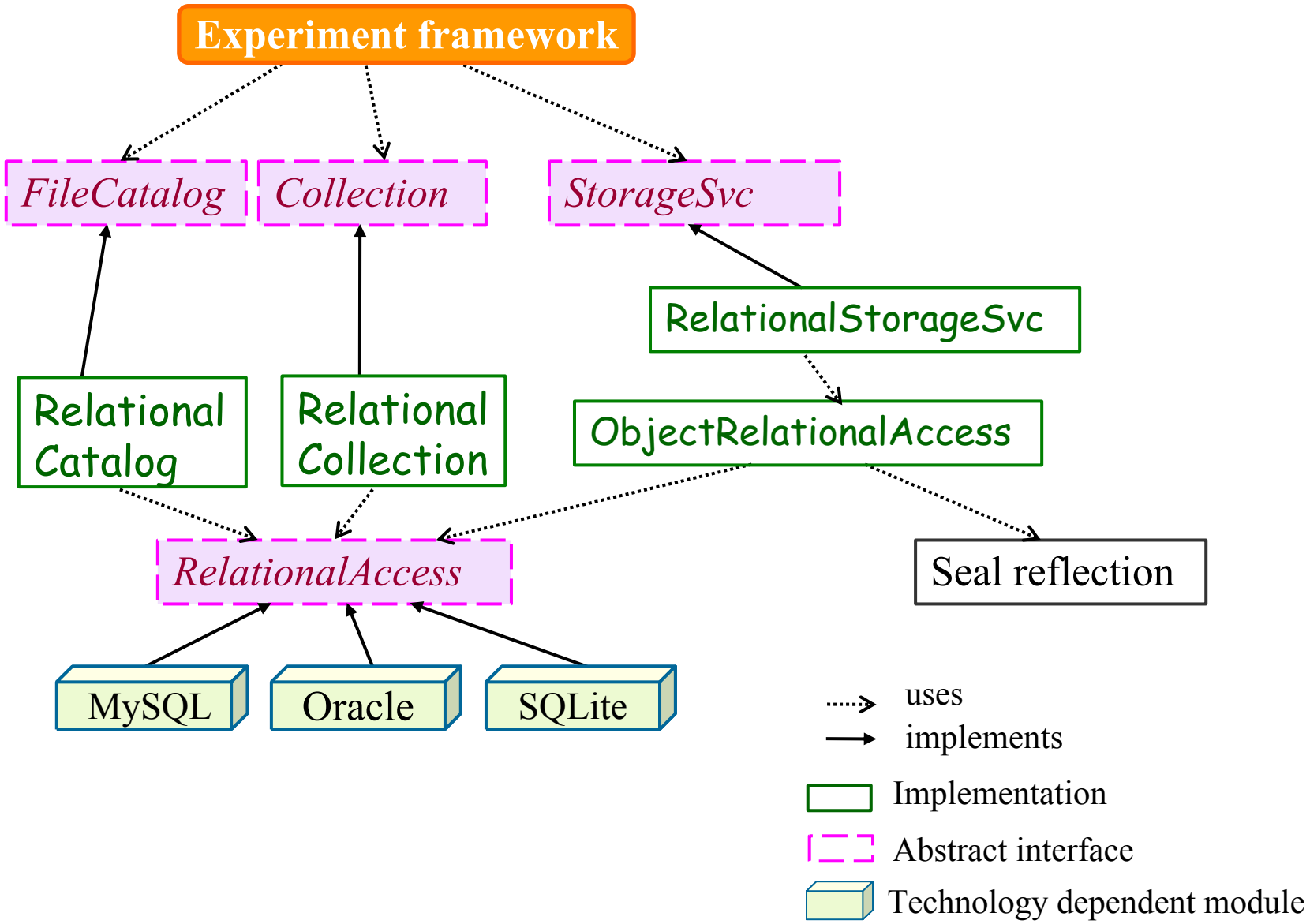
The software project (II)

- Drafted a project plan
  - ...which was too optimistic in terms of human resources
- Manpower
  - Developers: Zhen, Radovan, Giacomo, Ioannis
    - None working full time
  - Beta-testers: Vakho, Michael,...
  - Input on design strategy: Andrea, Markus, Dirk,...

Domain decomposition

1. Pure relational data management
   - Provide technology neutral RDBMS connectivity and SQL-free data management
   - Clients: file catalog, collection, conditions IoV
2. Object-relational mapping and storage
   - Bridge the differences between relational and object concepts (object identity resolution, object associations)
   - Provide guided object storage
   - Client: POOL Relational Storage Service
3. POOL Relational Storage Service
   - Adapter implementing the POOL StorageSvc interfaces
   - Client: experiment framework

Software design



1st domain:
The Relational Abstraction Layer
(R.A.L.)

Design concepts in RelationalAccess

- Design of public interfaces driven by the software requirements
- AttributeList interface used for the handling and the description of the relational data
- Only C++ (no SQL) types exposed
  - Type converters responsible for default and user-defined type conversion.
- SQL fragments only in the WHERE clauses of Queries and DML.

Technology plugin management

- Based on the SEAL component model
  - No SEAL plugin management specifics exposed.
  - Simplifies implementation and the collaboration among the plugins.
- User only has to deal with the POOL RelationalService
  - Automatic plugin loading based on the connection string.
  - User code based entirely on the abstract interfaces

Connecting to a database

- Connection string format:

technology_protocol://hostName:portNumber/databaseOrSchemaName:sidNumber

- Examples:

oracle://dbhost/user

mysql://dbhost:1105/dbname

sqlite_http://dbhost/directory/dbfile.db

sqlite_file:/absolute_dbfile_path.db

- No authentication parameters
- Connection string specifying only the data, not the access mechanism
  - The RelationalService decides which plugin to use.
  - Current convention: loading of the module named "POOL/RelationalPlugins/technology"

Authenticating

- Explicitly, specifying the user name and password values
- Implicitly, via an IAuthenticationService
  - explicitly provided
  - registered in the SEAL context tree
- IAuthenticationService:
  - Given a connection string, provides the connection parameters.
  - Two existing implementations (plugins):
    - environment variables (ignores connection string)
    - XML file with connections and parameters

RelationalAccess functionality

- Basic schema handling
  - Describing existing and creating new tables
  - Support for primary, foreign keys and indices (composed by one or more columns)
- Data Manipulation Language
  - Inserting, deleting, updating rows
  - Bulk row insertions
- Queries
  - Nested queries involving one or more tables
  - Ordering and limiting the result set
  - Control of client cache for the result set
  - Scrollable cursors

RDBMS plugins

- Oracle
  - Based on OCI 9.2.0.4
  - Fully supports the R.A.L. interfaces
  - Available for the Linux platforms (win32 will follow)
- SQLite
  - A light-weight embeddable SQL database engine
  - File-based (zero configuration, administration)
  - Available for the Linux and Win32 platforms
- MySQL
  - Implementation based on the MyODBC driver
  - Work in progress...

First clients of the R.A.L.

- RelationalFileCatalog
  - Validated the functionality and semantics of the interfaces
  - Tested against Oracle and SQLite
- Eager beta-testers from the experiments
  - CMS online
  - ATLAS detector geometry

2nd domain:
The Object-Relational mapping and object storage in RDBMS

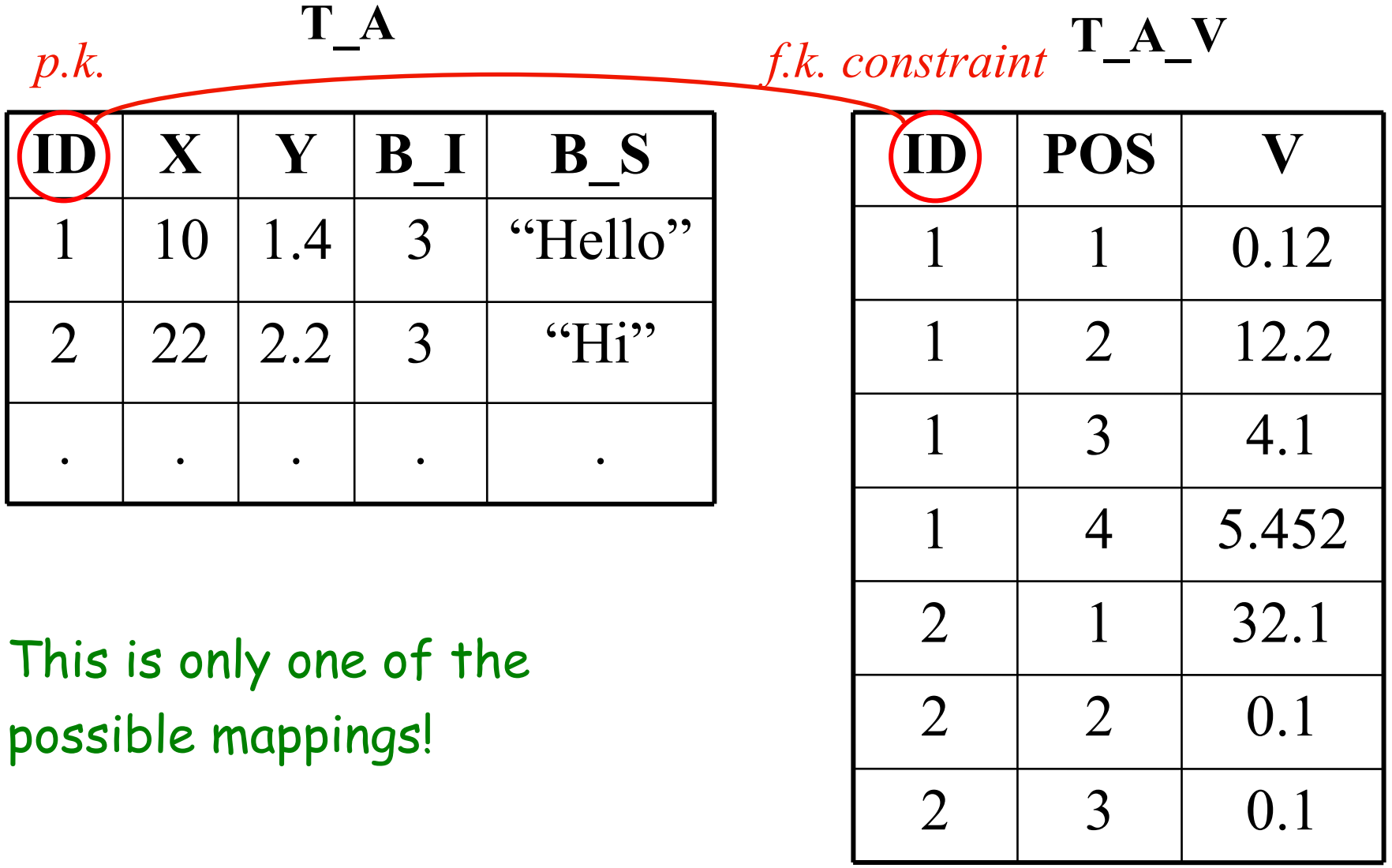
Basic concepts behind object storage in a relational database

- Classes ↔ Tables
  - both describe data layout
  - no unique mapping
  - need to store mapping together with the object data
  - need to store mapping versions
- Object identity (persistent address)
  - requires unique index for addressable objects
  - part of mapping definition

Mapping example (I)

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

Mapping example (II)



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

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!

Mapping Elements

- A mapping :
  - Version
  - Hierarchical tree of mapping elements
- An element:
  - Element type ("Object", "Primitive", "Array", "POOL reference", "Pointer")
  - Associated table name
  - Associated variable name
  - Associated variable type
  - Associated columns
  - Associated mapping elements
- Everything can be stored in 3 relational tables

Mapping generation

- Prerequisites :
  - C++ class(es) already defined
    - A tool will be provided for the automatic generation of C++ header files given a schema.
  - The SEAL dictionary libraries already generated
- Tool will be provided for the user-driven mapping generation:
  - XML input file to
    - Select the C++ classes
    - Override default mapping rules
    - Define the mapping version
  - Mapping gets "materialized" and stored in the database

Guided object storage

- Object I/O via the ObjectRelationalPersistency interface
  - For every object I/O operation the client has to supply:
    - the corresponding SEAL dictionary for the object's class
    - the object/relational mapping
    - the "persistent address" (eg. the value of the primary key in the table corresponding to the object's class)
  - Object data stored/retrieved following the SEAL dictionary information, and finding the corresponding entries in the mapping
    - Many schema evolution cases can be treated transparently through this mechanism

3rd domain:
The POOL Relational Storage Service

An adapter as a plugin for the POOL Storage Service

- A POOL "object":
  - Mapping version
  - Value(s) of indexed parameter(s)
- A POOL "container"
  - A table holding the values of the "object" structure
- A POOL "database"
  - Oracle user schema
  - MySQL database
  - SQLite file

Status and Plans

- First version of the R.A.L. already available with POOL 1.7.0
  - RelationalAccess, AuthenticationService, OracleAccess, SQLiteAccess
  - First client: RelationalFileCatalog
  - MySQLAccess being implemented
- ObjectRelationalAccess in progress
  - Mapping persistency implemented and tested
  - Mapping materialization implemented and partly tested
  - Object storage being implemented
  - Related tools defined but not yet be implemented
- RelationalStorageSvs in progress
  - Basic design ready
  - Related tools to be defined and implemented

What is still required

- Software upgrades of underlying packages:
  - AttributeList
    - Support for all primitive C++ types
    - Define and support a Blob type
    - Package needs some clean up
- Conventions
  - Format of the connection string
    - Can we agree on an LCG-wide convention?
- Manpower
  - Currently only few developers
  - More feedback (test cases) from the experiments
    - Both for the R.A.L. and the Storage Manager.

…to conclude

- The project is active with the first deliverables already available.
- Some documentation is already available from
  - the POOL User Guide
    http://pool.cern.ch/releases/POOL_1_7_0/doc/UserGuide
  - The RelationalAccess component description
    http://pool.cern.ch/releases/POOL_1_7_0/doc/RelationalAccess/RelationalAccess
- A lot of work is still ahead of us…