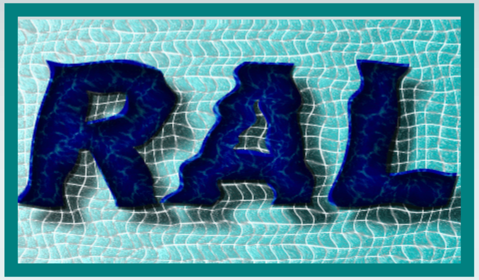
The POOL
Relational Abstraction Layer



3D Workshop
CERN, December 2004

Radovan Chytracek
CERN/IT/DB - LCG

Outline

- Introduction
- POOL architecture & RAL
- Features
- Example
- Common status & per-plug-in status
- Relational File Catalog
- Issues
- New developments
- Conclusions

Introduction

- **Motivation: independence from DB vendors**

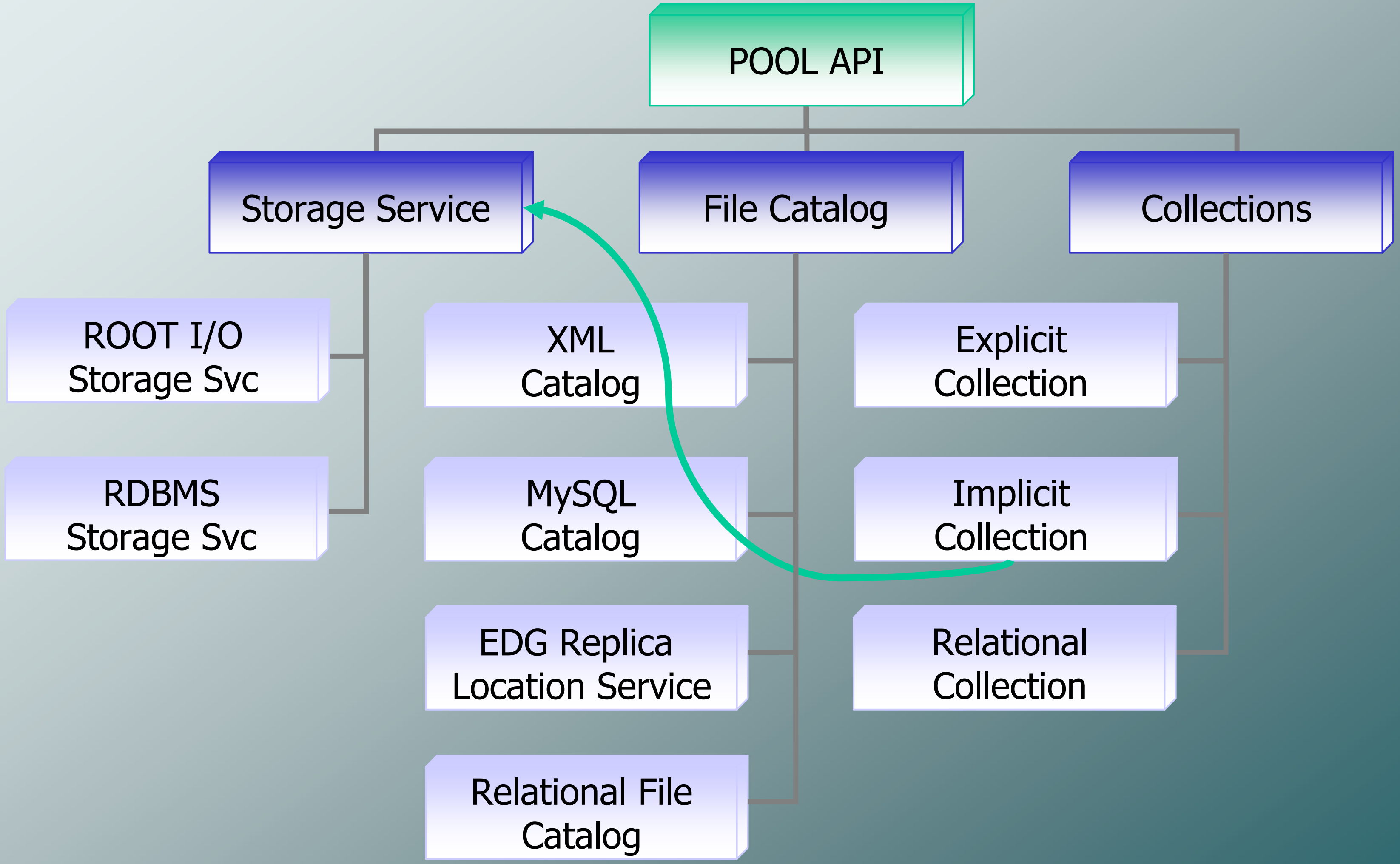
- **Activity started for most parts only in March.**
  - Requirements collection
  - Domain decomposition
  - Draft project plan

- **Addressing the needs of the existing POOL relational components (FileCatalog, Collection), the POOL object storage mechanism (StorageSvc) and eventually also the ConditionsDB (if requested by the experiments).**

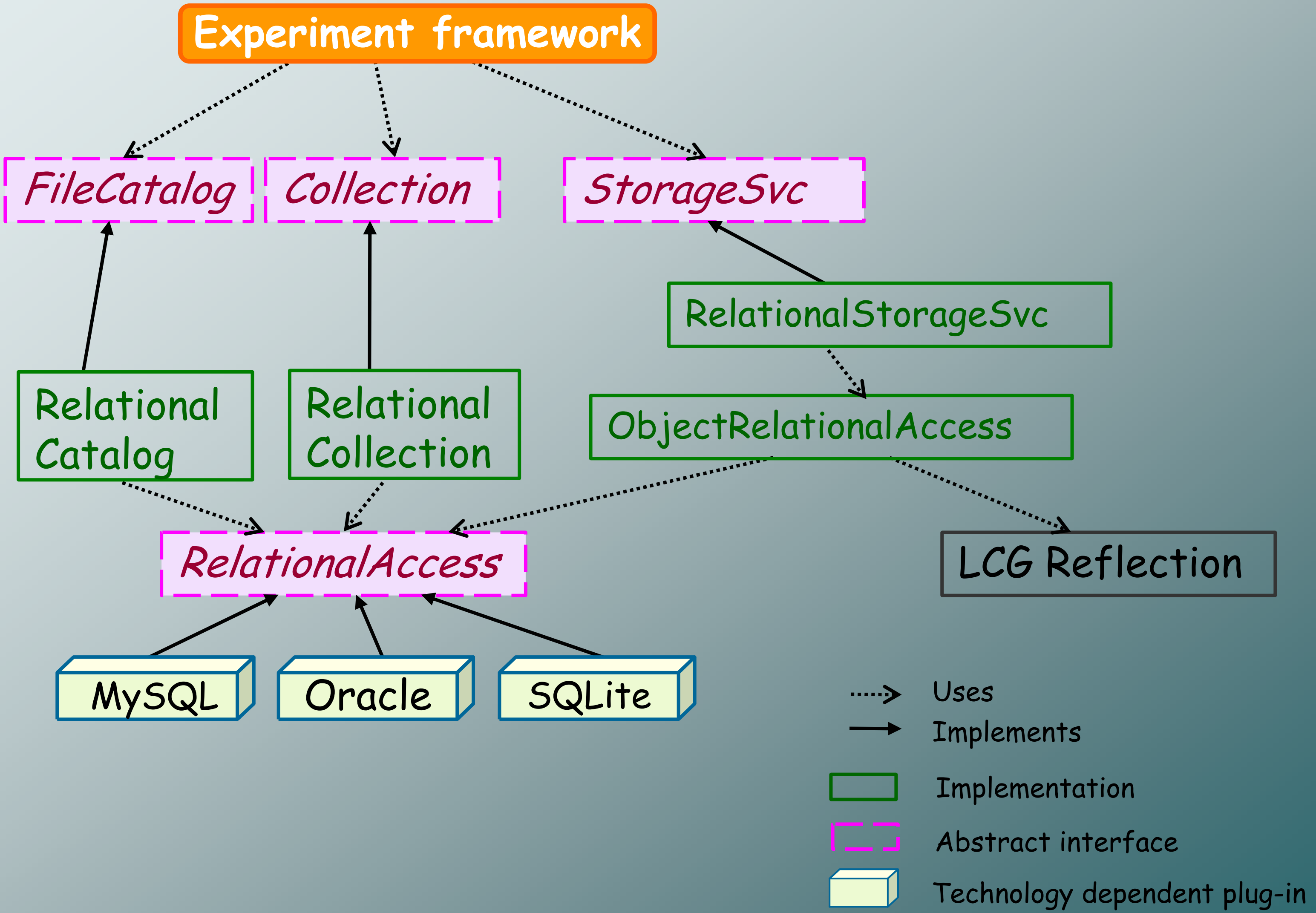
- **The use-cases and requirements are defined and updated in close cooperation with experiments**

- **Main developers:**
  Ioannis Papadopoulos, Zhen Xie,
  Radovan Chytracek, Giacomo Govi

POOL components



in POOL


Experiment framework
FileCatalog
Collection
StorageSvc
RelationalStorageSvc
Relational Catalog
Relational Collection
ObjectRelationalAccess
RelationalAccess
LCG Reflection
MySQL
Oracle
SQLite
Uses
Implements
Implementation
Abstract interface
Technology dependent plug-in

Features

- **Abstract, SQL-free API**
  - With exceptions of WHERE & SET clauses
- **Connection strings storable in a file catalog**
  - Example: mysql://raltest/RAL
  - Design decision: no connection credentials in the connection string
- **Schema, table, constraints & index handling**
  - DDL and meta-data functionality
- **Variable binding**
  - Named variables syntax supported, e.g. :VARNAME
  - ODBCAccess plug-in accepts positional ?-syntax as well
- **Queries against single or multiple tables**
  - Left joins possible
  - Sub-queries (back-end dependent)
- **Cursors**
  - Scrollable
- **Bulk inserts**
  - Emulated if not supported by the back-end client API or server

Domain decomposition

- Database access
  - IRelationalService, IRelationalDomain, IRelationalSession, IAutheticationService
- Schema handling
  - IRelationalSchema, IRelationalTable, IRelationalTableDescription, IRelationalTableSchemaEditor, IRelationalTableIndexEditor, IRelationalIndex, IRelationalPrimaryKey, IRelationalForeignKey, IRelationalTablePrivilegeManager, IRelationalTypeConverter
  - AttributeList
- Queries
  - IRelationalQuery, IRelationalSubQuery, IRelationalQueryWithMultipleTable, IRelationalCursor, IRelationalTableDataEditor, IRelationalBulkInserter
- Transactions
  - IRelationalTransaction

Example - Connection

```
POOLContext::loadComponent("POOL/Services/XMLAuthenticationService" );
POOLContext::loadComponent("POOL/Services/RelationalService" );

seal::IHandle<IRelationalService>
  serviceHandle = POOLContext::context()->
     query<IRelationalService>("POOL/Services/RelationalService");

IRelationalDomain& domain = serviceHandle->
    domainForConnection("mysql://raltest/RALTEST");

std::auto_ptr<IRelationalSession>
  session(domain.newSession("mysql://raltest/RALTEST"));

session->connect();

session->transaction().start();
session->userSchema().dropTable( "DataTable" );
session->transaction().commit();
```

Example - Create Table

```
session->transaction().start();

std::auto_ptr<IRelationalEditableTableDescription>
  desc( new RelationalEditableTableDescription( log, domain.flavorName()));

desc->insertColumn("id", AttributeStaticTypeInfo<int>::type_name());
desc->insertColumn("x", AttributeStaticTypeInfo<float>::type_name());
desc->insertColumn("y", AttributeStaticTypeInfo<double>::type_name());
desc->insertColumn("c", AttributeStaticTypeInfo<std::string>::type_name());

IRelationalTable&
  table = session->userSchema().createTable( "DataTable", *descr );

session->transaction().commit();
```

Example – Insert Data

```
session->transaction().start();

IRelationalTable& table = session->userSchema().tableHandle("DataTable");

AttributeList data( table.description().columnNamesAndTypes() );

IRelationalTableDataEditor& dataEditor = table.dataEditor();

for ( int i = 0; i < 5; ++i ) {
  data["id"].setValue<int>( i + 1 );
  data["x"].setValue<float>( ( i + 1 ) * 1.1 );
  data["y"].setValue<double>( ( i + 1 ) * 1.11 );

  std::ostringstream os; os << "Row " << i + 1;
  data["c"].setValue<std::string>( os.str() );

  dataEditor.insertNewRow( data );
}

session->transaction().commit();
```

Example - Query

```
// Querying : SELECT * FROM DataTable WHERE id > 2
std::auto_ptr<IRelationalQuery> query( table.createQuery() );
query->setRowCacheSize( 5 );

AttributeList emptyVarList;
query->setCondition( “id > 2”, emptyVarList );

IRelationalCursor& cursor = query->process();

if(cursor.start()) {
 while(cursor.next()) {
  const AttributeList& row = cursor.currentRow();
  for( AttributeList::const_iterator iCol = row.begin();iCol != row.end(); ++iCol ) {
   std::cout << iCol->spec().name() << " : " << iCol->getValueAsString() << "\t";
  }
  std::cout << std::endl;
 }
}

std::cout << “Selected row(s):" << cursor.numberOfRows() << std::endl;

session->transaction().commit());
session->disconnect();
```

Common Status

- The latest is POOL release POOL_1_8_2-alpha
  - First RAL components available since POOL 1.7.0
- Base interfaces defined
  - Strictly following requirements
- AuthenticationService implementations available:
  - XML and shell environment based
- Oracle, ODBC/MySQL and SQLite plug-ins
  - unit-tested and excercised by ObjectRelational StorageService
- Proof of concept RelationalFileCatalog implemented
  - tested with Oracle, SQLite and MySQL servers
- First implementation tag of RelationalCollections
  - Yesterday ☺

Oracle plug-in

- **Oracle plug-in**
  - Uses Oracle OCI C API
  - Based on Oracle 10g
    - Supports connection to 9i and 10g servers
    - Makes use of the “binary_float” and “binary_double” SQL types
  - Can be used with the Oracle 10g instant client
- **Status**
  - Fixed all known bugs and introduced CLOB support

SQLite plug-in

- Flat file database engine
  - Tiny memory footprint
  - Understands most of SQL-92
  - Easy to use API
- First implementation based on SQLite version 2
  - File size and variable binding issues
- Now based on SQLite version 3
  - File size went down by factor of 2
  - Real variable binding implementation in progress

MySQL Status

- **MySQL access is via ODBC**
  - ODBC-based implementation
  - Native implementation now would run into maintenance problems as MySQL API is changing through versions 4.0 to 4.1 to 5.1
  - Until 5.1 is out POOL access to MySQL via the more generic ODBC plug-in will be kept
- **Uses UnixODBC + MyODBC 3.51**
  - Native ODBC manager on Windows
- **Tested against MySQL 4.0.18+**
- **MySQL server requirements**
  - InnoDB and ANSI mode are required to keep the RAL semantics

Relational File Catalog

- **Generic, RAL-based implementation of the FC interfaces**
  - RAL proof of concept
  - Exists since POOL 1.7.0
  - In testing now by CMS
    - SQLite, Oracle

- **Scalability/performance tests using the Oracle & MySQL/ODBC plug-in**
  - RLS CMS production data as testing sample
    - ~3x10^6 entries + metadata (~1x10^6)
  - RAL based replication test:
    - Oracle: 90 minutes (dual CPU node, 2GB RAM)
      - True bulk inserts
    - MySQL: 10 hours (single CPU node, 1.25.GB RAM)
      - No bulk inserts in MySQL 4.0.x API used by MyODBC
      - Good speed up to 10^6 entries

Relational Collection

- First tag release notes ☺
  - Developed by Ioannis Papadopoulos and Kristo Karr

- Multiple collections in a given database/schema
- No restriction in the collection and variable names
  - as they do no longer map directly to table and column names
- Protection from concurrent writters through row locking
- Use of links tables for the efficient storage of the tokens
- Provision for future extensions in the ICollection interface such as
  - retrieving the collection size
  - removing of records
  - addressing individual records
  - retrieving the list of the referenced databases/containers.

Issues

- **Nested queries problems with ObjectRelational StorageService**
  - SQLite & MySQL/ODBC (under investigation)
- **CLOB trap when using bulk inserts**
  - '\0' bytes not truncated by MySQL for TEXT columns
  - to be fixed in MySQL & checked for Oracle plug-in
- **MySQL 4.0.x InnoDB does not scale well over 10^6 entries**
  - Perhaps due to single shared table space file
  - We'll see in 4.1.7 where table space-per-table is possible
  - TEXT column type to be used with care
    - Storage overhead + slow query speed

New Developments

- **Will review soon the existing interfaces**
  - Extension of the table description interface (column size)
  - Support of BLOB types and "long long"
- **MySQL 4.1.7 native plug-in trial**
  - Still no cursors in 4.1, binary protocol & variable binding is +
  - Easy migration with MyODBC 3.53 for MySQL 4.1.7
    - Available by January 2005
- **RelationalCollections**
  - First prototype is available
  - Testing and integration with real collection data (ATLAS)
- **ODBCAccess plug-in re-factoring**
  - Allow support for more RDBMs: Oracle, PostgreSQL
  - Most of the points of variability already analyzed
  - Low priority

Conclusions

- We did it ☺
  - Coding started in March – full implementations by now
- Oracle plug-in works in all cases
- SQLite & MySQL plug-ins in 99%
- All back-ends heavy stressed by POOL ObjectRelational StorageService
  - see the next talk by Ioannis Papadopoulos
- RAL successfully used in implementationa across all POOL application domains
  - File catalog, Collections, StorageService
- Our Thanks to CMS developers and ATLAS geometry database team for close collaboration and useful feedback