Persistence for Analysis Objects

J.Hřivnáč*, LAL, Orsay, France

Abstract

There are two kinds of analysis objects with respect to their persistence requirements:

- Objects, which need direct access to the persistence service only for their IO operations (read/write/update/...): histograms, clouds, profiles, ... All Persistence requirements for those objects can be implemented by standard Transient-Persistent Separation techniques like JDO [1], Serialization, etc.
- Objects, which need direct access to the persistence service for some of their standard operations: NTuples, Tags,.... It is not feasible to completely separate Transient and Persistent form of those objects. Their Persistence should be tightly interfaced with their transient form. One possibility is to directly implement a persistent extension of those objects for each persistence mechanism.

The SQLTuple[1] has been developed to deliver efficient SQL persistence for AIDA [2] standard NTuple objects. The implementation is based on FreeHEP [3] AIDA implementation and is completely inter-operable with other FreeHEP components as well as with other AIDA implementations. SQLTuple dependency on SQL database implementation is handled at run-time by textual configuration. In principle all mainstream SQL databases are supported. The default mapping layer can be customized so that, for example, LCG [4] Pool [5] Tag databases can be transparently supported. This customization is used to implement higher level management utilities for Pool Tag databases - package ColMan. ColMan[2] utilities are accessible also from the C++ environment and via standard Web Service.

PERSISTENCE

Persistence Classification

There are two kinds of AIDA objects with respect to persistence requirements (see Figure 1):

- Objects, which are read/written entirely in one step: IHistograms, IClouds, IProfiles, ... All Persistence requirements for those objects can be implemented by standard persistence techniques based on Transient-Persistent Separation (JDO, Serialization,...)

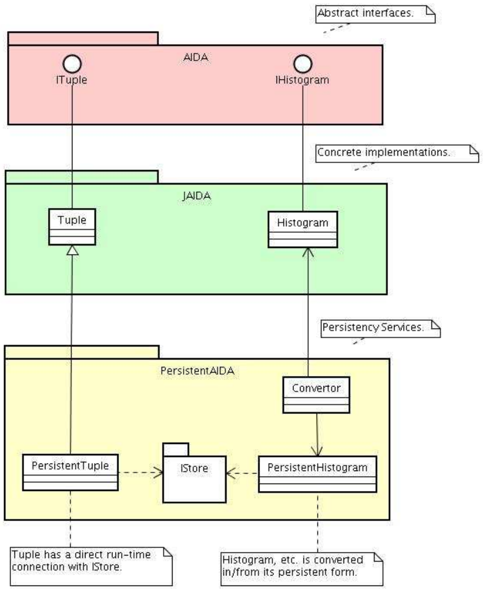

Figure 1: Two kinds of Analysis Objects.

- Objects, which are read/written only partially in one step and objects which are only interrogated: ITuples. It is not feasible to completely separate Transient and Persistent form of those objects. Their Persistence should be tightly interfaced with their transient form. One possibility (chosen here for SQL-based persistence) is to implement an ITuple extension for each persistence mechanism.

ORTHOGONAL PERSISTENCE

Java Data Objects [1]

JDO (see Figure 2) is currently the mainstream persistence technology for Objects. JDO implementations (both free and commercial) exist for practically all existing Databases (relational or not).

All JDO Enhancers are compatible.

JDO provides all standard database functionality (transaction, caching,...).

JDO and AIDA [2]

FreeHEP AIDA can be persistified using Java Data Objects (JDO).

- FreeHEP AIDA storage API (IStore) should be enhanced to support real database (the current one sup-

* Julius.Hrivnac@cern.ch

[1] http://home.cern.ch/hrivnac/Activities/Packages/SQLTuple

[2] http://home.cern.ch/hrivnac/Activities/Packages/ColMan

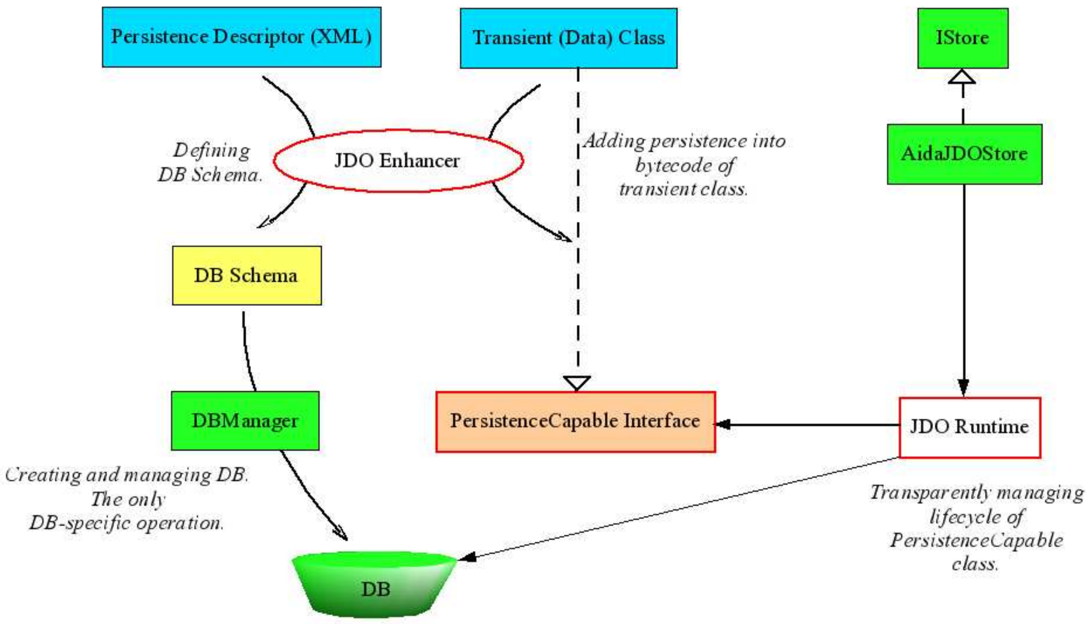

Figure 2: Java Data Objects.

ports only file-like API).

- AIDA objects can be decoupled from ITree management.
- More powerful query machinery can be introduced.

AIDA specifies Interfaces, but persistence should work on Data. There are two ways to handle it:

- Persistify current FreeHEP implementation of AIDA. This solution is
  - easy to implement,
  - faster,
  - not portable (depends on actual FreeHEP implementation).
- Create AIDA DataClasses (from AIDA XML Schema by XSLT StyleSheets) and persistify them.
  - Transient-Persistent converters should be written to convert between DataClasses and actual implementation of AIDA. Non-intrusive Aspects can be used as converters.
  - It is slower.
  - It is portable (AIDA XML Schema is common to all AIDA implementations).

Following is a simple example of AIDA code using JDO persistence:

```
/** Plot all 1-dim histograms
  * from a database. */

// Start AIDA
IAnalysisFactory af = IAnalysisFactory.create();
IPlotter page =
  af.createPlotterFactory().create("Page");
page.show();

// Start JDO
PersistenceManager pm =
  Connection.getPM(databaseProperties);
Transaction tx = pm.currentTransaction();
tx.begin();

// Get Histograms
Query query = pm.newQuery(Histogram1D.class);
Collection result = (Collection)query.execute();
page.createRegions(result.size() / 2 + 1, 2);

// Use Histograms
Iterator it = result.iterator();
int i = 0;
while (it.hasNext()) {
  page.region(i++).plot((IHistogram1D)it.next());
  }

// Close JDO
tx.commit();
pm.close();
```

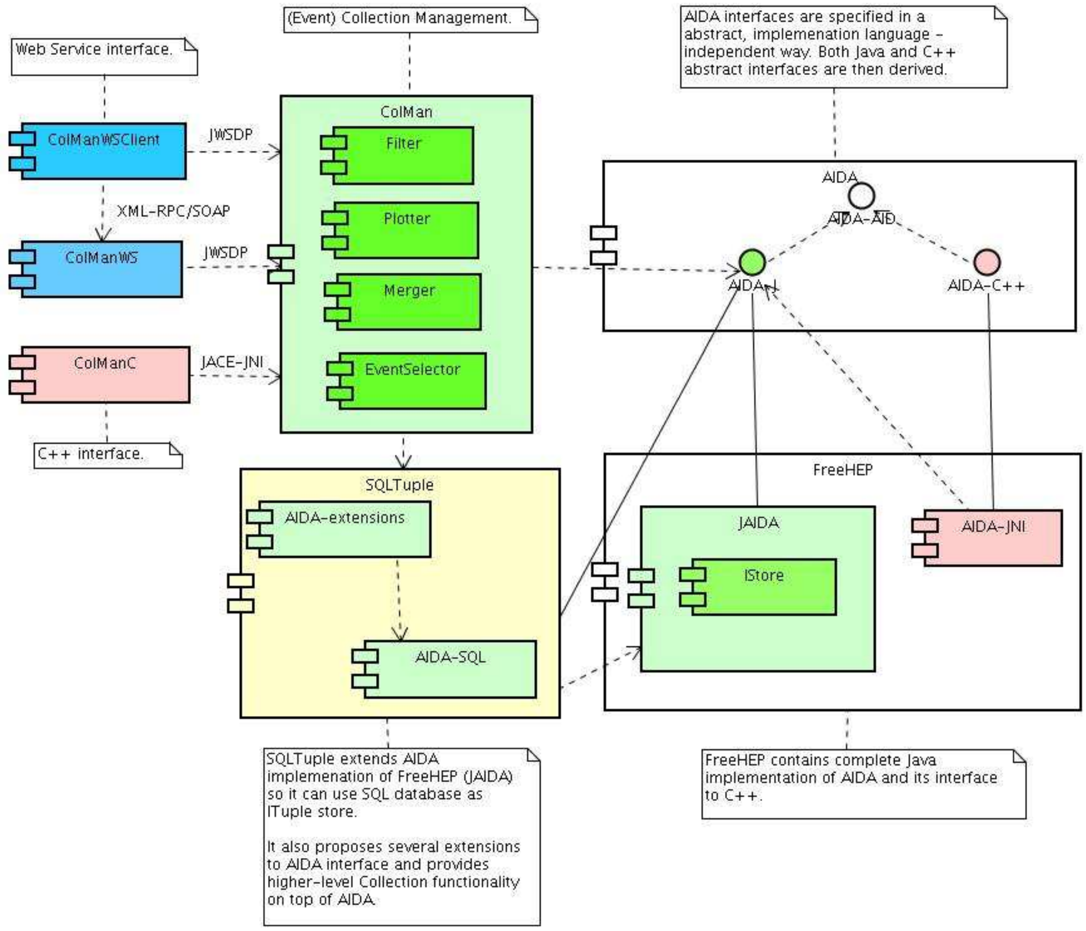

Figure 3: Java Data Objects.

DIRECT PERSISTENCE

SQLTuple and ColMan Component Diagram is shown on Figure 3.

SQLTuple

SQLTuple implements AIDA interface as an extension of the FreeHEP JAIDA implementation. ColMan contains several extensions (Filter, Plotter, Merger, EventSelector,...) using AIDA to perform global operations on NTuples (not only SQLTuples). All the functionality is available using standard AIDA interfaces, WebService access and JACE [6]-created C++ proxies.

SQLTuple implementation of AIDA interface (including its extensions) is in hep.aida.ref.sql package. Higher level extensions are in net.hep.atlas.Database.Collections.Management package.

NTuple Persistence

AIDA NTuples can be stored using many different storage technologies (Compressed XML files, Root [7] files, HBook files, several SQL databases).

Operations between technologies (filtering, merging,...) are possible via standard AIDA channels. Within SQL technology, native SQL channels are used to speed up operations.

COMPARISONS

Advantages over alternative Tools

SQLTuple runs on any platform without recompilation and can be compiled on any platform using any Java compiler version 1.5+. Distribution compatible with Java 1.4 can be provided if needed.

All Relational Databases can be supported in local (embedded) or remote (server) mode (as long as such modes are supported by the database).

The performance of SQLTuple is in most cases higher than performance of equivalent C++ implementation.

SQLTuple can be easily used from other languages, like Python [8], Ruby [9], Groovy [10], PNuts [11] or C++ [12] or as a language-neutral Web-Service.

All SQL mapping (both types and commands) is customizable via text files.

Added Values for LCG Pool

Standard API, already used in many Application Frameworks and well know to Users.

Platform-independent, multi-language API.

Many AIDA tools ready to access/process metadata.

Support for wide range of RDBSs.

Global operations (chaining, merging, filtering,...) on NTuples.

Added Values for AIDA

SQL databases to store NTuples.

COMPONENTS

SQLTuple [20]

SQLTuple extends FreeHEP implementation of ITuple AIDA interface so that ITuples can be stored in an SQL database. It supports any relational DB backed via JDBC [13] interface (configuration is provided for MySQL [14], PostgreSQL [15], McKoi [16] and Oracle [17], basic tests have been performed also for Cloudscape [18] and Hypersonic [19]). All AIDA operations (projections, filters, evaluators,...) are supported in a standard way. Some new functions have been included on top of standard AIDA Interface.

The implementations is, in principle, ready to be used in any AIDA-complaint tool.

SQLTuple can be used to access and manage LCG Pool Event Metadata SQL storage.

SQLTuple depends on SQL only via (textual) run-time configuration files:

- **Implementation.properties** describes generic properties of SQL backends (protocol name, JDBC driver, capabilities,...).
- **Type.properties** specifies SQL-Java type mapping (for all involved SQL backends).
- **StmtSrc.properties** defines SQL commands to be used to perform AIDA functions (like ITuple.project(...), etc.).

ColMan [21]

ColMan (Collection Management) provides higher-level functionality for the management of Event-level metadata (Collections).

It supports LCG Pool metadata (SQL fully, Root files in read mode). Following functions are available:

- Filtering - creation of subCollections based on selection string (Query).
- Merging - concatenation of several Collections into one.
- Replicating - copying Collections into different technologies and/or sites.
- Plotting - creation of AIDA Histograms and Clouds from Collections.
- Selecting - looping over Event entries selected by a Query.

ColMan functionality is exported to other languages and Frameworks via:

- ColManC - C++ proxies to ColMan classes.
- ColManWS - XML-RPC Web Service.

REFERENCES

[1] JDO - Java Data Objects (http://access1.sun.com/jdo, http://www.jdocentral.com)

[2] AIDA - Abstract Interfaces for Data Analysis (http://aida.freehep.org)

[3] FreeHEP - HEP Components and Tools for Java (http://java.freehep.org)

[4] LCG - LHC Computing Grid Project (http://lcg.web.cern.ch/LCG)

[5] Pool of Persistency Objects for LHC (http://lcgapp.cern.ch/project/persist)

[6] JACE - C++ Interface to Java (http://sourceforge.net/projects/jace)

[7] Root - Object Oriented data Analysis Framework (http://root.cern.ch)

[8] Python - Interpreted, interactive, object-oriented programming language (http://www.python.org)

[9] Ruby - Interpreted scripting language for quick and easy object-oriented programming (http://www.ruby-lang.org)

[10] Groovy - Agile dynamic language for the JVM (http://groovy.codehaus.org)

[11] Pnuts - Script language for Java environment (https://pnuts.dev.java.net)

[12] C++ - Low level compiled programing language with some Object Oriented features (http://www.cplusplus.com)

[13] JDBC - Java Database Connectivity (http://java.sun.com/products/jdbc)

[14] MySQL - SQL Database (http://www.mysql.com)

[15] PostgreSQL - SQL Database (http://www.postgresql.org)

[16] McKoi - SQL Database (http://mckoi.com/database)

[17] Oracle - SQL Database (http://www.oracle.com)

[18] Cloudscape - SQL Database (http://www-306.ibm.com/software/data/cloudscape)

[19] Hypersonic - SQL Database (http://hsqldb.sourceforge.net)

[20] SQLTuple - AIDA ITuple with SQL Backend (http://home.cern.ch/hrivnac/Activities/Packages/SQLTuple)

[21] ColMan - (Event) Collection Management for AIDA ITuples (http://home.cern.ch/hrivnac/Activities/Packages/ColMan)