Optimized access to distributed relational database system

J.Hrivnac, LAL, Orsay, France

Abstract

Efficient and friendly access to the large amount of data distributed over the wide area network is a challenge for the near future LCG experiments. The problem can be solved using current standard open technologies and tools. A JDBC standard solution has been chosen as a base for a comprehensive system for the relational data access and management. Widely available open tools have been reused and extended to satisfy HEP needs.

- An SQL backend has been implemented for the Abstract Interface for Data Analysis (AIDA), making relational data available via standard analysis API. Interfaces to several languages, plugins to Java Analysis Studio as well as Web Service access are available and interfaced with Atlas Event Metadata (Tag) database.
- Clustered JDBC (Sequoia) from ObjectWeb Consortium has been reused to enable transparent and optimized access to distributed relational database. Several extensions are under development.
- Octopus replication tool from ObjectWeb Consortium has been extended with specific requirements of the Atlas experiment and used to replicate Atlas data over heterogeneous database network.

JDBC

Java Database Connectivity (JDBC)[1] is the foundation of almost all Java database application. It can access most relevant SQL databases. Enormous amount of high-quality applications (free or commercial) is based on JDBC. JDBC allows to construct a complete databased-based framework with a minimum additional code.

SQLTUPLE AND COLMAN

SQLTuple[2] extends FreeHEP[3] implementation of ITuple AIDA[4] interface so that ITuples can be stored in an SQL database. It supports any relational DB backend via JDBC standard interface. All AIDA operations (projections, filters, evaluators,...) are supported in a standard way. Some new functions have been included on top of standard AIDA Interface. SQLTuple can be used in any AIDA-compliant tool. SQLTuple understands LCG[5] Pool[6] AttributeList data. ColMan[7] provides higher level utilities for managing (Event) Collections. See Fig. 1 for SQLTuple and ColMan Design Overview.

JAS Plugins

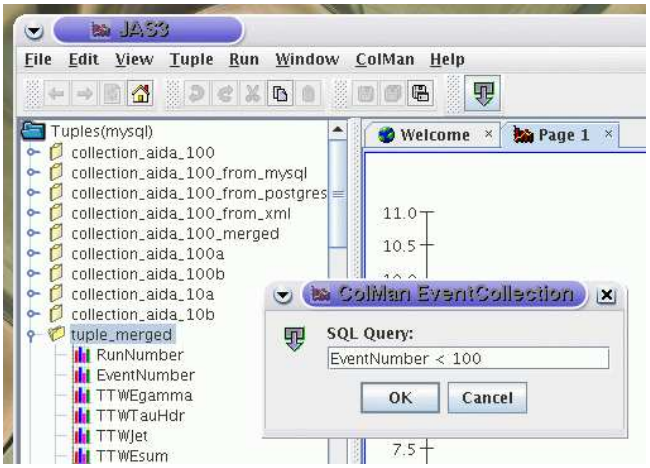

Figure 2: JAS Plugins.

Both SQLTuple and ColMan are available as JAS[8] plugins (see Fig. 2). All functionality is available via standard JAS GUI. Specific functionality (like SQL queries) is added to the JAS GUI. Interoperability with other JAS plugins is assured.

Multilanguage Interfaces

Selected SQLTuple/ColMan functionality is available to legacy C++ applications via proxies created by JACE[9]. All SQLTuple/ColMan functionality is available to Python applications using Jython[10] or JPype[11] (when access to C++ is needed too). Access from other languages (Ruby[12], Groovy[13],...) is possible directly.

Web Service

ColMan Event Selector is available via XML-RPC[14] service (see Fig. 3). The service is generally deployed as */ColMan/EventSelector* Web Service application and can be accessed via ColMan Web Service Client. JSP (Java Server Pages)[15] GUI for ColMan tools has been deployed as */ColMan-GUI* Web Service application and can be accessed directly from the standard Web Browsers.

SEQUOIA

SQL tables can be spread over several database Servers, some tables may be replicated. User wants a single front-end. Sequoia[16] (described on Fig. 4) acts as a (Proxy)

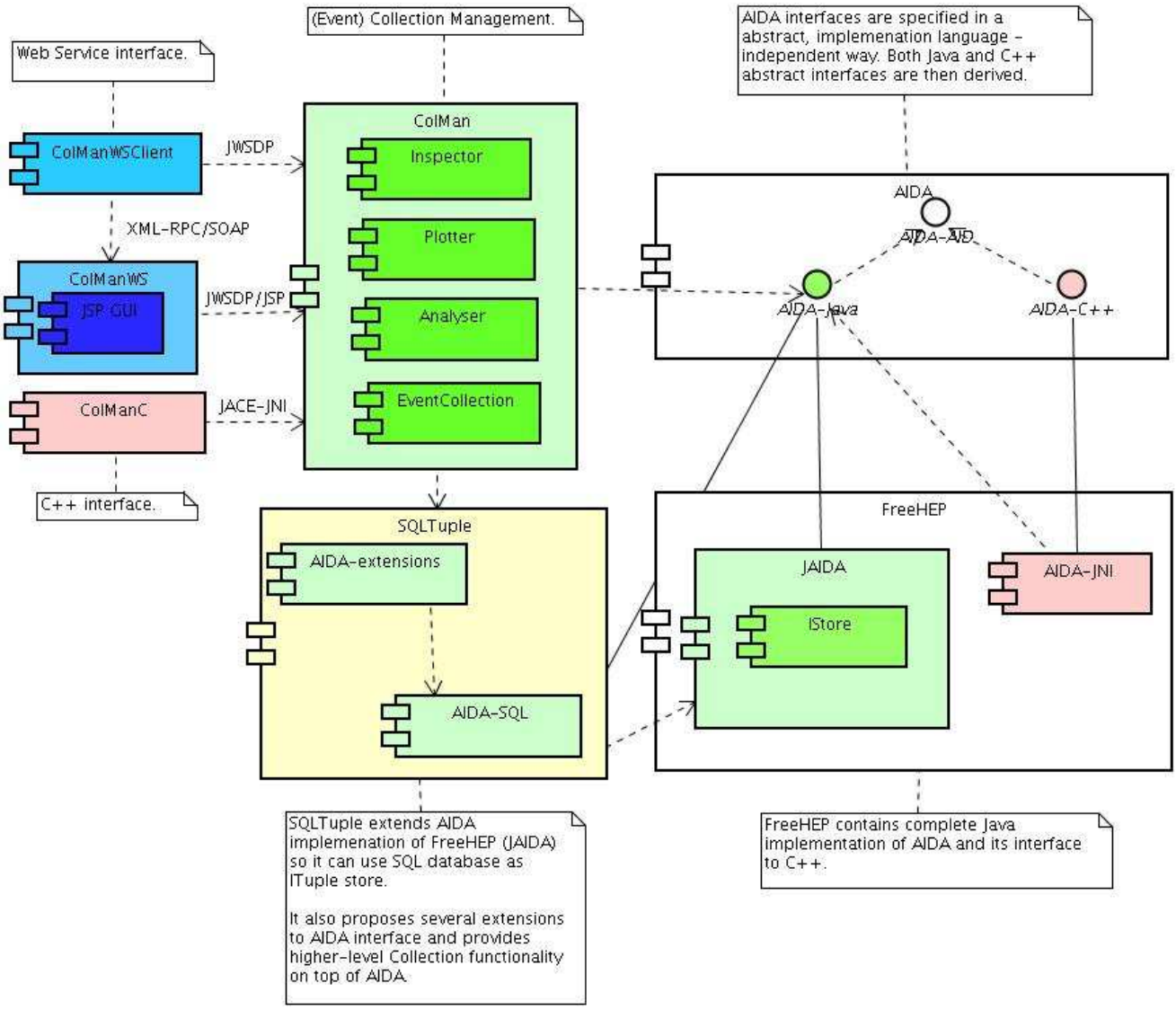

Figure 1: SQLTuple and ColMan.

Virtual SQL Server forwarding all requests to appropriate databases (real or another virtual). Replicated and/or complementary tables are supported (even on heterogeneous Servers), similar to RAID disks. Sequoia is used via its JDBC driver, so any application using JDBC API can directly use Sequoia. No application modification is required to use Sequoia. Sequoia directly handles any SQL query. No pre-knowledge is needed, no specialized interface is required. Sequoia handles both query (read access) and update (write access). JDBC interface can be transparently accessed from other languages (Python, Ruby, Groovy, C++,...). Sequoia has been developed by Continuent[17].

OCTOPUS

Octopus[18] (described on Fig. 5) is a Java-based Extraction, Transformation, and Loading tool. It may connect to any JDBC data sources and perform transformations defined in an XML file. Octopus has been customized to support non-standard SQL features used in LCG and to overcome LCG-specific bugs. Octopus is routinely used to replicate Atlas Detector Description database. It has been successfully tested with other Atlas databases. Octopus has been developed by Enhydra[19].

REFERENCES

[1] http://java.sun.com/products/jdbc

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

[3] http://java.freehep.org

[4] http://aida.freehep.org

[5] http://lcgapp.cern.ch

[6] http://lcgapp.cern.ch/project/persist

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

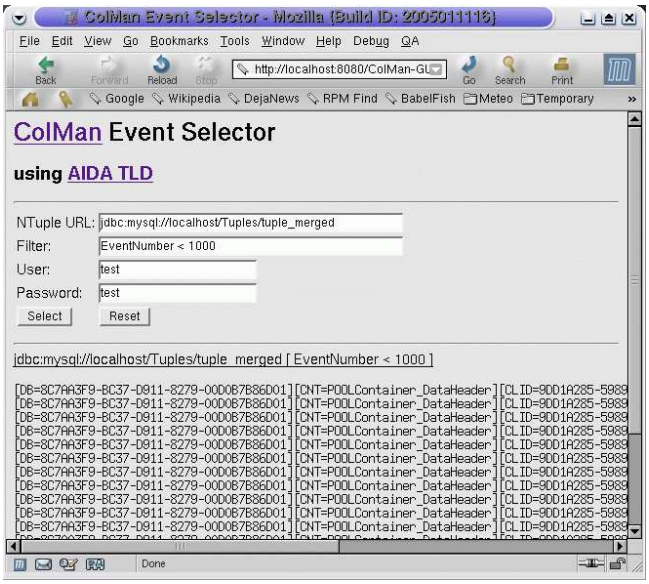

Figure 3: Web Service.

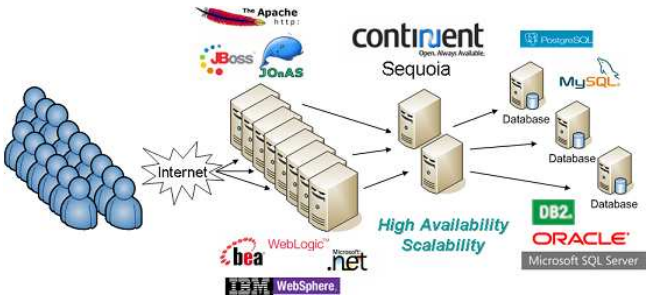

Figure 4: Sequoia.

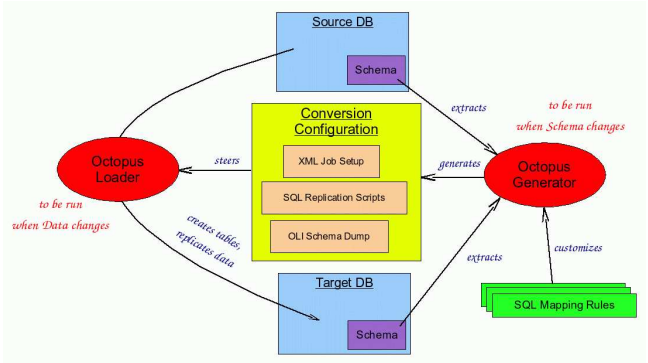

Figure 5: Octopus.

[8] http://jas.freehep.org/jas3

[9] http://sourceforge.net/projects/jace

[10] http://www.jython.org/

[11] http://jpype.sourceforge.net/

[12] http://www.ruby-lang.org

[13] http://groovy.codehaus.org

[14] http://www.xmlrpc.com

[15] http://java.sun.com/products/jsp

[16] http://sequoia.continuent.org/HomePage

[17] http://www.continuent.org

[18] http://www.enhydra.org/tech/octopus

[19] http://www.enhydra.org