Event Collections as AIDA ITuples

SQLTuple + ColMan

- Requirements on Event Collections API
- Architecture:
  - Requirements on Event Collections
  - Technology Choices
  - SQLTuple Architecture
  - FreeHEP AIDA Storage
  - Proposed SuperAIDA
- Applications:
  - ColMan – Collection Management
  - Web Service
  - Metadata Management
  - Pool Compatibility
  - Python and C++ Interface
  - Metadata Analysis
- Performance
- What's new
- Summary

SQL storage
for FreeHEP implementation
of ITuple AIDA interface
compatible with Pool Event Collections Metadata

```
http://hrivnac.home.cern.ch/hrivnac/Activities/Packages/SQLTuple
http://hrivnac.home.cern.ch/hrivnac/Activities/Packages/ColMan
http://java.freehep.org
http://aida.freehep.org
http://lcgapp.cern.ch/project/persist/metadata
```

Requirements on Event Collections
(Event MetaData, Tag DB, AttributeList)

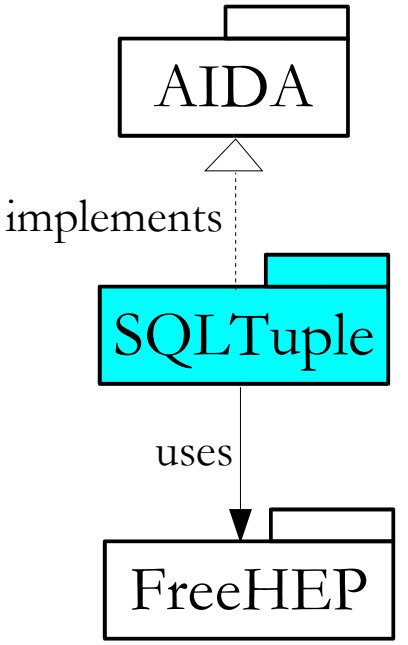
- **Collections Management** functionality should be provided (replications, filtering, merging, splitting,...).
- **Collections Navigation** functionality should be provided (searching, looping,...).
- **Analysis** functionality should be available (histogramming, combining, cutting,...).
- All functionality should be available in **multi-language** environment (Java, C/C++, Python,...).
- All functionality should be available in a **platform-independent** environment.
- API should be reusable with **other storage technologies** (SQL databases, XML files, Root files,...).
- **SQL syntax** should be **hidden**, user should use her native environment (Java, C/C++, Python,...).
- **SQL functionality** should be **used** (performance, advanced functions when available,...).
- **Any** SQL **database** should be supported (MySQL, PostgreSQL, Oracle, embedded DBs,...).
- **Distributed** (Grid) environment should be possible (WebService,...).
- Compatibility with **Pool** Event Metadata should be possible.
- **Performance** overhead over native SQL should be negligible.

Technology Choices

- **AIDA** for API:
  - Event Metadata are AIDA ITuples.
- **FreeHEP** for AIDA implementation:
  - It is the most complete and functional AIDA implementation.
  - It supports most storage technologies.
  - It can be used from Java, C++, Python and PNuts.
- **JDBC** for access to SQL databases:
  - It satifies all requirements.
  - It is the most widely accepted API.
  - It is supported for all common RDBS.
- **WebService** for distributed and multilanguage interface:
  - It is the only real standard.
- **Java** for native implementation
  - It has all needed properties and functionality.
  - It can be easily used from other languages.
  - It runs everywhere (no porting problems).
- **JACE** for interface to C++:
  - It makes using Java from C++ easier that another C++ from C++.

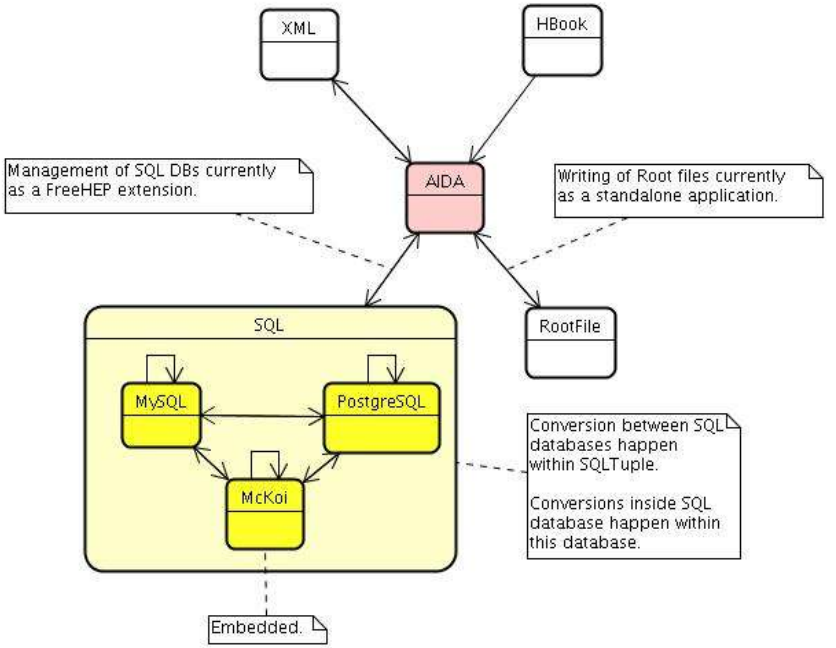
SQLTuple Architecture

- SQLTuple extends existing FreeHEP implementation of AIDA by implementing SQL IStorage.
- It uses the same API as other AIDA FreeHEP storage technologies, like XML or Root files.
- It is very simple (there are just two classes with non-trivial mission).
- All dependencies on SQL and database-specific features are stored in textual configuration files:
  - *Implementations.properties* for database-dependent properties,
  - New *Types.properties* for SQL-Java types mapping,
  - *StmtSrc.properties* for SQL-ITuple methods mapping.
- SQLTuple provides several extensions to existing AIDA interfaces (proposed as AIDA standard):
  - Extensions to ITuple interface:
    - Richer access methods.
    - SQL-aware methods (searching, indexing,...).
  - Extensions to IStore interface:
    - IStore API is not yet standardized.
    - Current IStore fits with file-based technologies (XML, RootIO,...), but not with real databases.
    - Only AIDA XML format is actualy standardized.

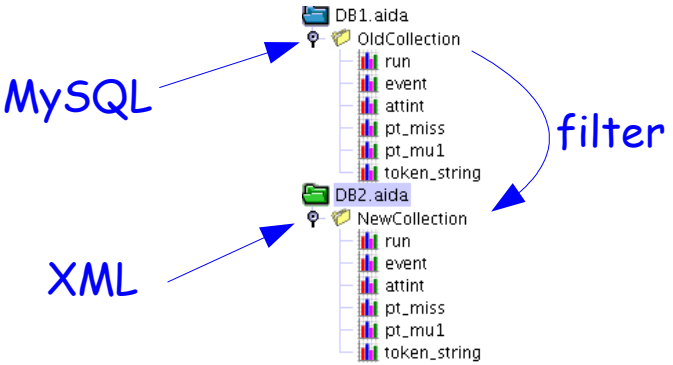


FreeHEP AIDA Storage

- AIDA ITuples can be currently stored in many technologies using FreeHEP AIDA implementation:
  - AIDA (compressed) XML files,
  - Root files:
    - Root TTuples (reading integrated in FreeHEP, writing via standalone prototype),
    - New: Pool Root AttributeLists (reading integrated in ColMan, writing not available as Pool Root files format hasn't been decrypted yet).
  - HBook files (only reading),
  - SQL databases (MySQL, PostgreSQL and McKoi directly supported; Cloudscape and Hypersonic tested):
    - Just tables,
    - New: Pool AttributeLists (with LinkTables).
  - ...

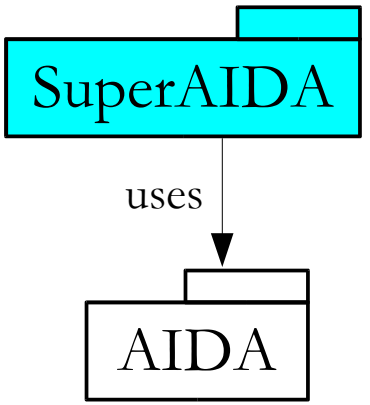


- All AIDA standard operations are supported.
- ITuples (=Collections) in ITree can be mounted, linked, copied, moved,... as in the (distributed) filesystem. They can be manipulated directly from the code (in many languages), using GUI or from the command line.
- All database operations profit from the native technology (e.g. copying of SQL ITuples is performed within SQL database where possible).



New Proposed SuperAIDA

- More complex AIDA operations on top of existing AIDA API to satisfy AttributeSet Use Cases:
  - no impact on current AIDA,
  - usable with any AIDA implementation.
- AttributeSet requirements:
  - ITuple specification by ITupleSpec object, itself serialisable into XML.
  - ITuple row accessible as ITupleEntry object (to fill or retrieve).
- Other requirements:
  - Support for vector and matrix columns.
- Other ideas:
  - Formal specification of AOD2AttributeSet relation ?




```
<AttributeSet name="..."
              version="..."
              author="...">
  <Description>
    ...
    </Description>
  <ScalarAttribute name="..."
                   type="..."
                   default="..."
                   comment="..."/>
  <VectorAttribute name="..."
                   type="..."
                   length="..."
                   default="..."
                   comment="..."/>
  <MatrixAttribute name="..."
                   type="..."
                   length1="..."
                   length2="..."
                   default="..."
                   comment="..."/>
  </AttributeSet>
```

ColMan - Collection Management

- **Filter** creates new (sub)Collection/(sub)Replica from existing one applying a filter to attributes (works between different storage technologies):

```
java -jar ColManFilter.exe.jar <inUrl> <outUrl> <filter> <inUser> <inPassed> <outUser> <outPasswd>
```

- **Merger** physically merges two Collections into new one:

```
java -jar ColManMerger.exe.jar <inUrl1> <inUrl2> <outUrl> ...
```

- **Plotter** plots selected attributes from Collection:

```
java -jar ColManPlotter.exe.jar <url> <x> <y> <weight> <filter> <user> <passwd>
```

New

- **Convertor** converts proprietary Pool-Root files into standard AIDA formats:

```
java -jar ColManPoolRoot2AIDA.exe.jar <inFile>.root [<outUrl>]
```

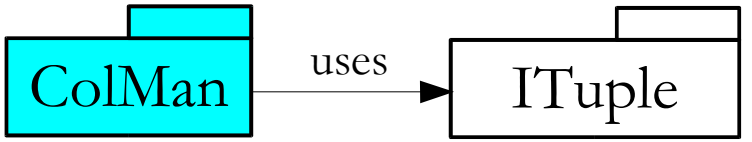
- **EventSelector** returns a set of Tokens of Events from Pool Collection using a filter on attributes:

```
EventCollection collection = new EventCollection(url, filter_string, user, passwd);
EventIterator iterator = collection.iterator();


String token;
while (iterator.next()) {
  token = iterator.token();
  // use token (find its Event,...) ...
  }

collection.close();
```

- All executable jar-files installed in InstallArea/share/lib.
- Just several examples, others can be easily added.
- Work across all supported storage technologies.
- Java and C++ direct API exist too.
- WebService interfaces exist.



Web Service

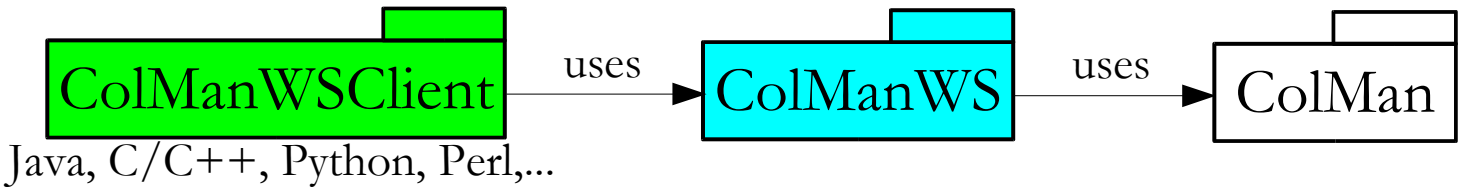
- WebService allows to access all the functionality from a very thin (remote) Client.
- ColMan utilities are exported using JWSDP WebService server.
- Other AIDA functionality can be easily added.
- Clients can be created from WSDL WebService descriptor in almost any language (even in C++).
- SQLTuple WebService can collaborate with other WebServices (like AMI), forming **Distributed Heterogeneous Collections Database**.

```
/** Web Client to SQLTuple EventSelector WebService.
  * All other code is created automatically from WSDL. */

// Get remote EventSelector
EventSelectorWS selector = new EventSelector_Impl().getEventSelectorWSPort();
selector.setProperty(ENDPOINT_ADDRESS_PROPERTY, "http://WebServer.there.net:8080/SQLTuple/EventSelector");

// Select set of Tokens
Object[] tokens = selector.select("jdbc:mysql://SQLServer.here.net/Tuples/TestCollection",
                                  "pt < 6",
                                  "user",
                                  "passwd");

// Loop over Tokens
for (Object token : tokens) {
  // use token (find its Event,...) ...
  }
```



New Metadata Management

- **Production:**
  1) Creation of Event files and Event Metadata files using Pool.
  2) Copying of Event files into their master storage.
  3) Merging of Event Metadata files into SQL Event Metadata database using ColMan, indexing.
  4) Registering of Event files in AMI and File Catalog.
  5) Registering of SQL Event Metadata in AMI.
  6) Replicating, ...
- **Processing/Analysis:**
  1) Selection of Collections from the Collections Metadata database using AMI ("*Give me all Higgs Collections.*")
  2) Selection of Event Tokens from the Event Metadata database using ColMan, possibly extraction into local database/file ("*Give me all Tokens for Events with pt > 6.*").
  3) Location of Events using Token interpreter and File/Replica Catalog, possibly extraction into local file ("*Where are my Events ?*").
  4) Extraction of Events from files using Pool, creating local copies ("*Give me my Events.*").
  5) Monitoring of usage patterns, re-indexing, replicating,...
- All data should be available in a distributed, language-neutral way using **WebServices:**
  - Collections Metadata (AMI).
  - Event Metadata (ColManWS).
  - Token interpreter (Pool doesn't provide its standalone incarnation yet).
  - File/Replica Catalog.
  - Event Server (delivering requested Events) or even Athena Service (performing submitted Algorithms wrapped as Agents) ?

New Pool Compatibility

Existing in SQLTuple+Colman to provide compatibility with Pool

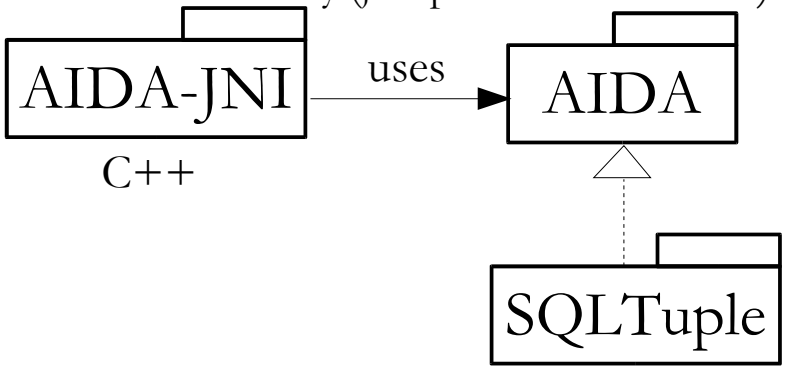
- SQLTuple+ColMan is **interoperable with Pool** SQL AttributeList.
- The default SQLTuple behavior is to map SQL table directly to AIDA ITuple.
- All SQL features (types, associated commands,...) are defined in a properties files, user can provide customized files which allow different mapping of SQL to ITuple (i.e. different Schema).
- Pool AttributeSets are stored in two tables: attributes themselves in one, Tokens in another (LinkTable):
  - Customized *StmtSrc.properties* file is used to access Pool AttributeSets.

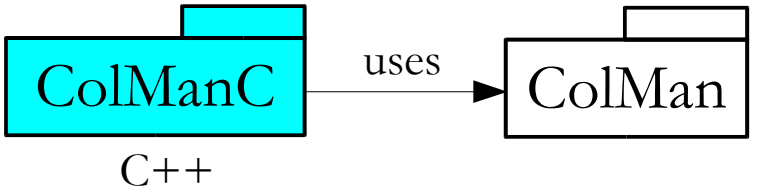
Missing in Pool to provide compatibility with AIDA

- Pool SQL storage (existing as well as planed) is not enough functional **to provide portability across applications and databases**:
  - Mapping between SQL and native types can't be independently specified and can't be deduced from the database content. Conservating reading is not possible without additional information.
  - Schema are not available without reading actual database.
  - Different database implementations don't use the same SQL dialect.
  - OIDs (Tokens) can't be interpreted outside Pool.
- Pool Collections should be customizable by external files:
  - SQL types and commands (to be) used to access data.
  - AttributeSet Specification (i.e. mapping to SQL types, default values, comments,...).
- Standalone Service for creation/interpretation of Tokens should be available.
- Standard formats of Pool Metadata should be supported:
  - AIDA XML,
  - Root TTuples (not standard, undocumented, but widely used and already decrypted).

Python and C++ Interface

- SQLTuple is written in Java (and SQL) to profit from mature infrastructure (JVM, JDBC, FreeHEP,...), largely un-available in other languages (C++).
- SQLTuple+ColMan can be used directly from **Python**.
- Many interfaces to **C++** are available:
  - FreeHEP AIDA implementation itself implements C++ AIDA interface via AIDA-JNI package (used, e.g., by Geant4; soon to be interfaced to PI). It can be used to access SQLTuple AIDA functionality.
  - Direct C++ proxies to ColMan utilities are created using JACE package (package ColManC).
  - ColMan JWSDP WebService (package ColManWS) can be transparently used by any WSDL C++ Web Client (AxisC, gSOAP,...).
- Other languages (**PNuts**, **Groovy**, **Ruby**,...) can be trasparently used too as they provide direct access to Java.
- Sizes (without externals):
  - SQLTuple library: 60 kB
  - ColMan library: 30 kB, ColMan executables: 400 B each
  - ColManC library (just proxies to ColMan !): 2.5 MB, ColManC executables: 200 kB each

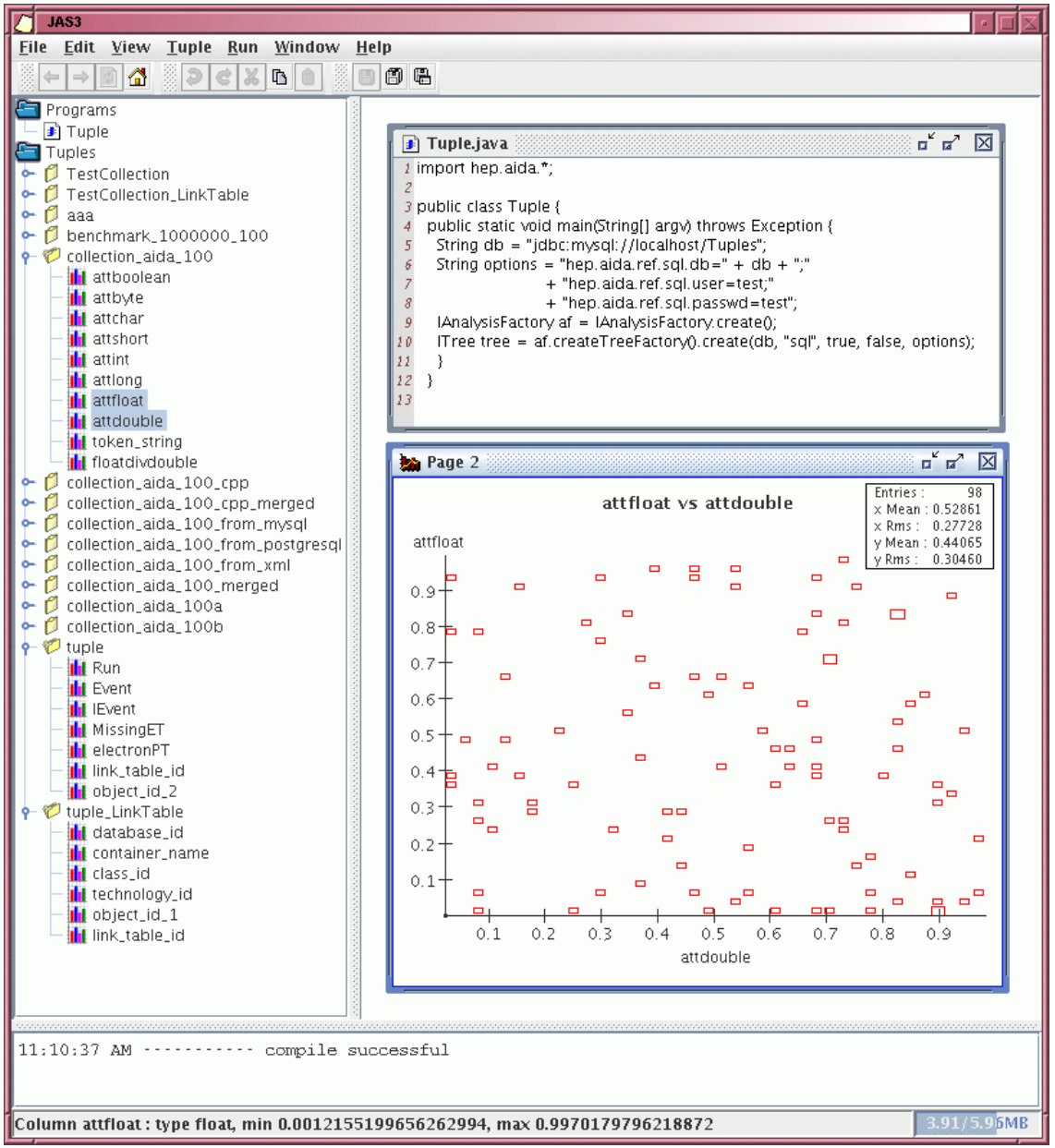


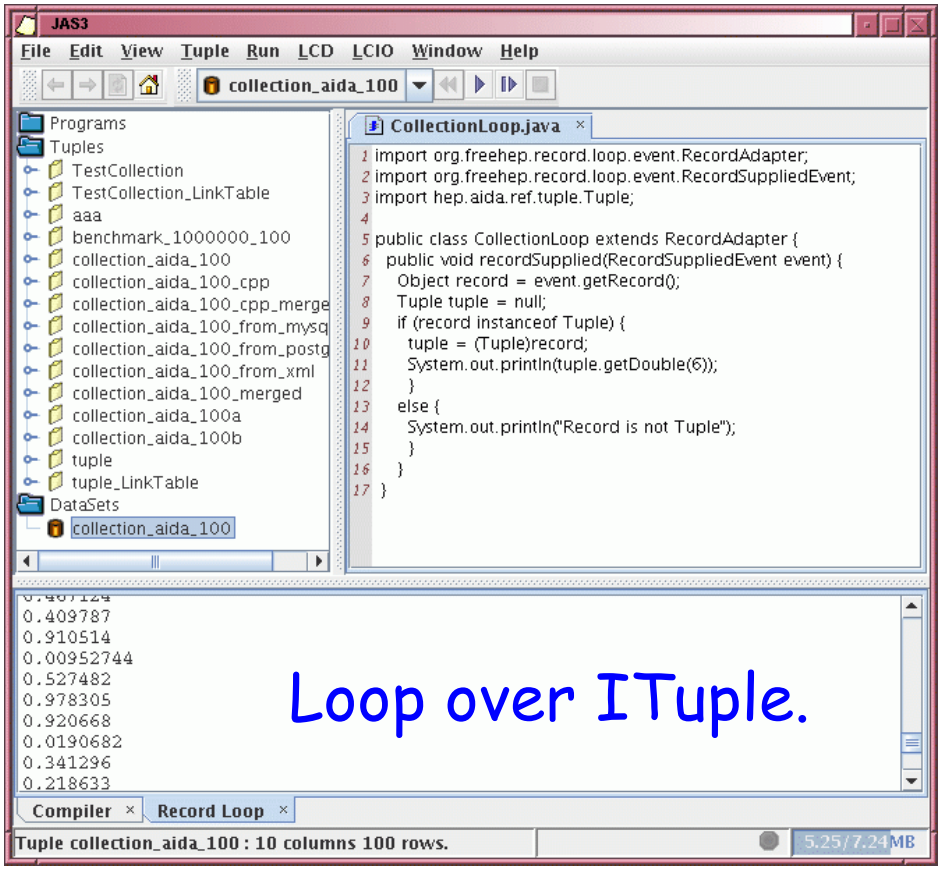


New

Metadata Analysis (With JAS + SQLTuple)

- Access to ITuple (SQL, Rootfile, XML files,...).
- Local or remote.
- Can analyze using any JAS facility:
  - Graphically,
  - By Java, Python or PNuts class/script,
  - From Python or PNuts command line.



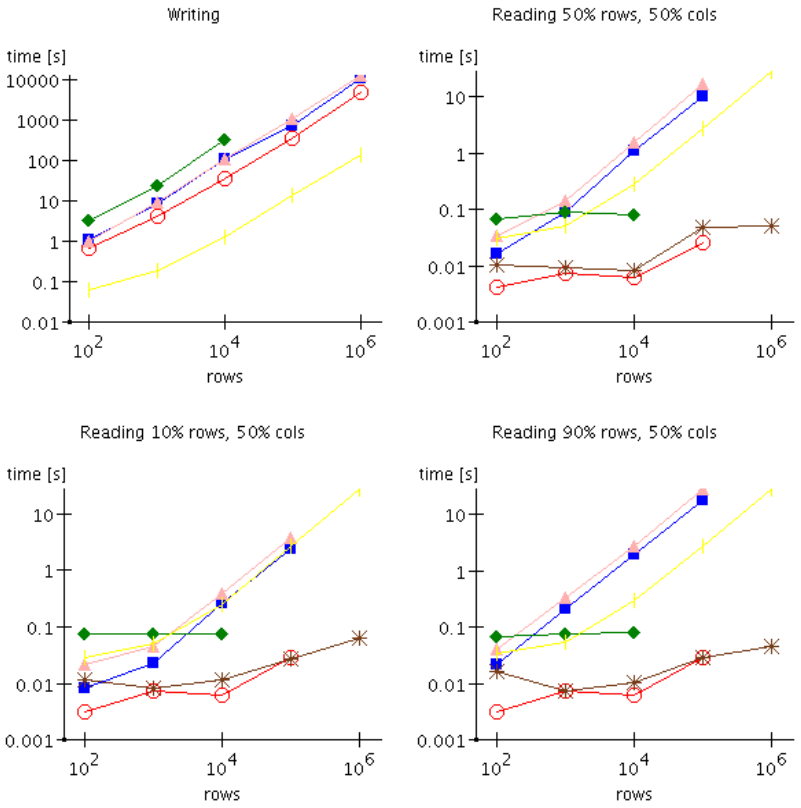


Loop over ITuple.

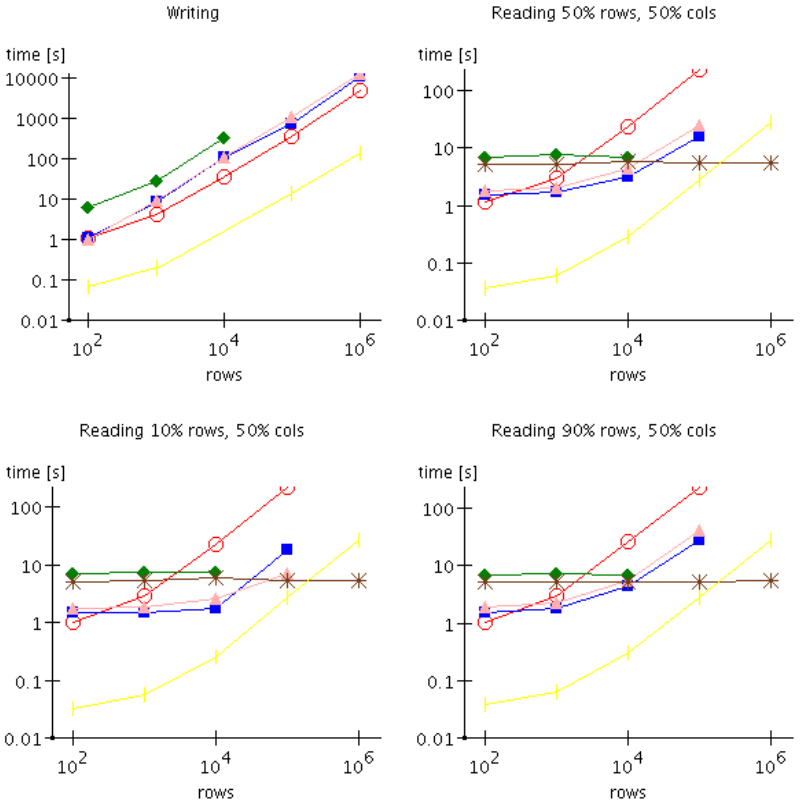
Performance

- Detailed CPU benchmarking suite is provided to evaluate all supported storage formats with different access patterns (reading subset of columns, selecting subset of rows by filter), results are available on the Web.
- However, comparison of different technologies can only be done on real applications in a real use because simplified benchmarks are always trapped by differences in optimization strategies:
  - Client-side and Server-side caches
  - Indexing
  - Hollow variables
  - Lazy loading
  - Dynamic optimization
  - Memory management
- Generally, CPU time needed to read an ITuple can be divided into two parts:
  - Constant overhead spent only once per ITuple, it is used to understand ITuple Schema, perform selections and prepare structures in memory.
  - Access time proportional to the amount of data read in.
- File-based, embedded and simple storage technologies seem to be more performant for flat access (when user reads everything or at least knows in advance what she will need).
- SQLTuple (and JDBC) brings in negligible overhead, most of the time is spend in low level access code.
- Size of stored ITuples depends linearly on number or entries and is (for 100000 rows * (50 floats + 50 ints)):
  - XML: 54MB
  - MySQL: 44MB
  - PostgreSQL: 21MB
  - McKoi: 223MB
  - Root: 42MB (25MB when compressed)

Performance - sample



Without constant overhead



With constant overhead

What's new
(Since Dec'03 SW WS)

- **Repackaging:** original SQLTuple split into SQLTuple, ColMan, ColManC, ColManWS and ColManWSClient.
- **Support for Pool AtributeSet:**
  - LinkTables in SQL,
  - Pool-Root files (reading).
- **No SQL in code:** all SQL dependency via textual configuration files.
- **SuperAIDA proposal:** additional AttributeSet functionaity on top of standard AIDA.
- **Wider availability:**
  - as standalone packages,
  - from Atlas release.
- **Ready for DC2:**
  - Collection Management utilities,
  - AttributeSet Analysis.
- **Usable in JAS.**

New things since Dec'03 presentation are marked as New

Summary

AIDA, extended by SQLTuple, is suitable API for Event Collections.
FreeHEP provides all necessary foundations.
ColMan is ready for DC2.

- **SQLTuple+ColMan** presents any SQL data as standard AIDA ITuples so it
  - can be transparently reused within any AIDA-compatible application,
  - can transparently reuse any other AIDA-based storage technology.
- **SQLTuple+ColMan** is platform independent (works automaticaly on any Linux, MS, MacOSX without recompilation).
- **SQLTuple+ColMan** supports any relational database (MySQL, Postgres and McKoi included, others tested)..
- **SQLTuple+ColMan** offers multi-language access (Java natively, Python directly, C++ via proxies, any language via WebService).
- **SQLTuple+ColMan** is compatible with Pool Event Metadata.
- **SQLTuple+ColMan** provides high level Collection Management Utilities.
- **SQLTuple+ColMan** can be used in a distributed (Grid) environment.
- **SQLTuple+ColMan** reuses: Java, FreeHEP, AIDA, JDBC, JACE, JWDSP, MySQL, PostgreSQL, McKoi, Log4J, Ant.
- **SQLTuple+ColMan** is available as a standalone package or from Atlas release.