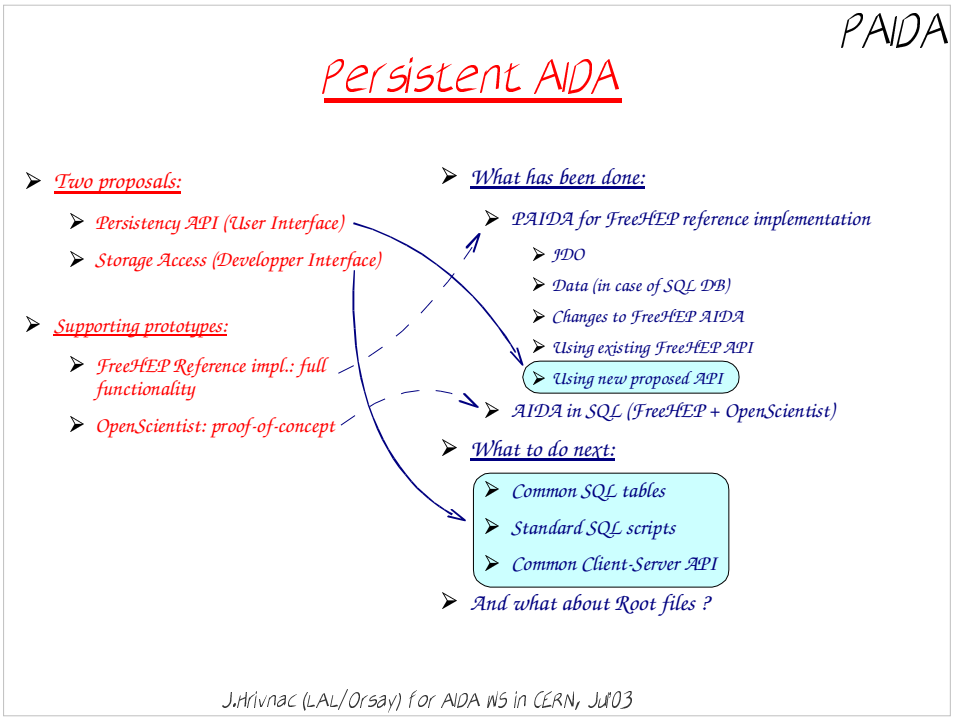
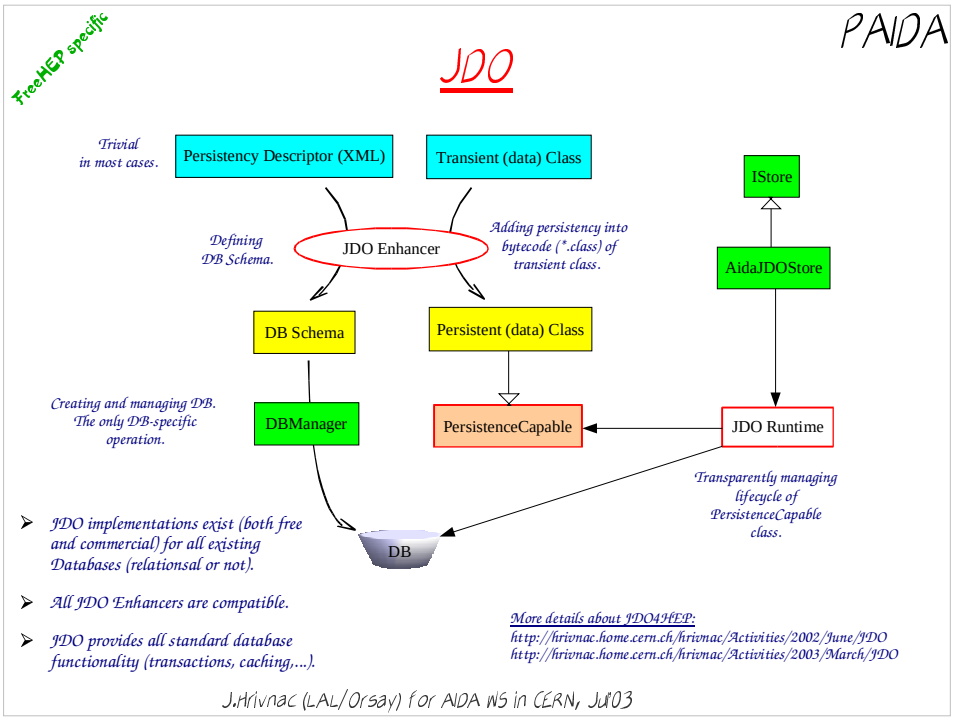
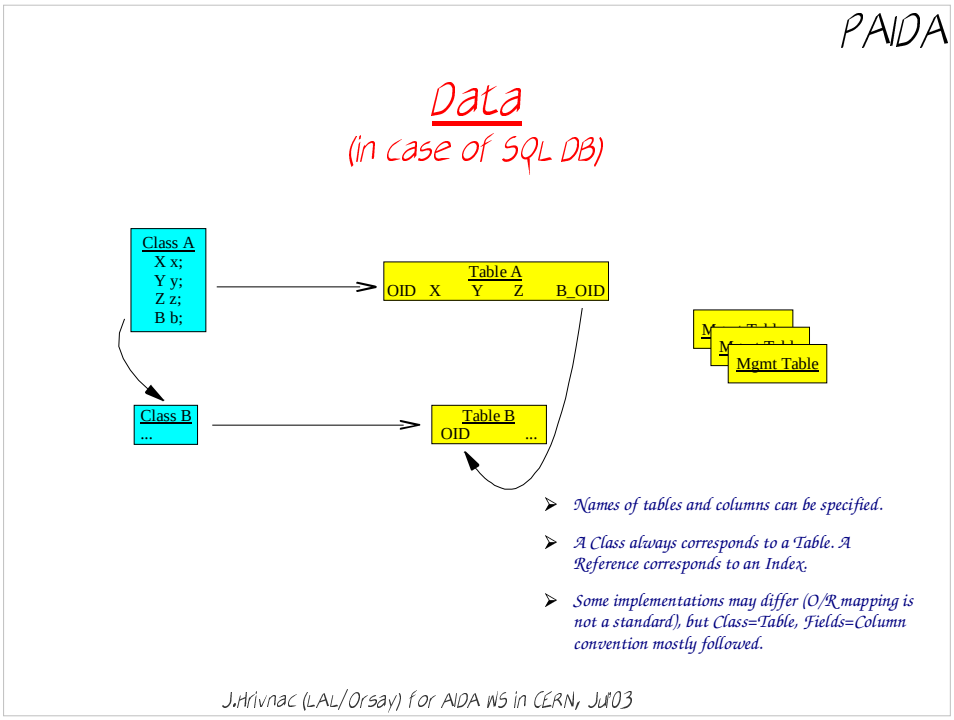


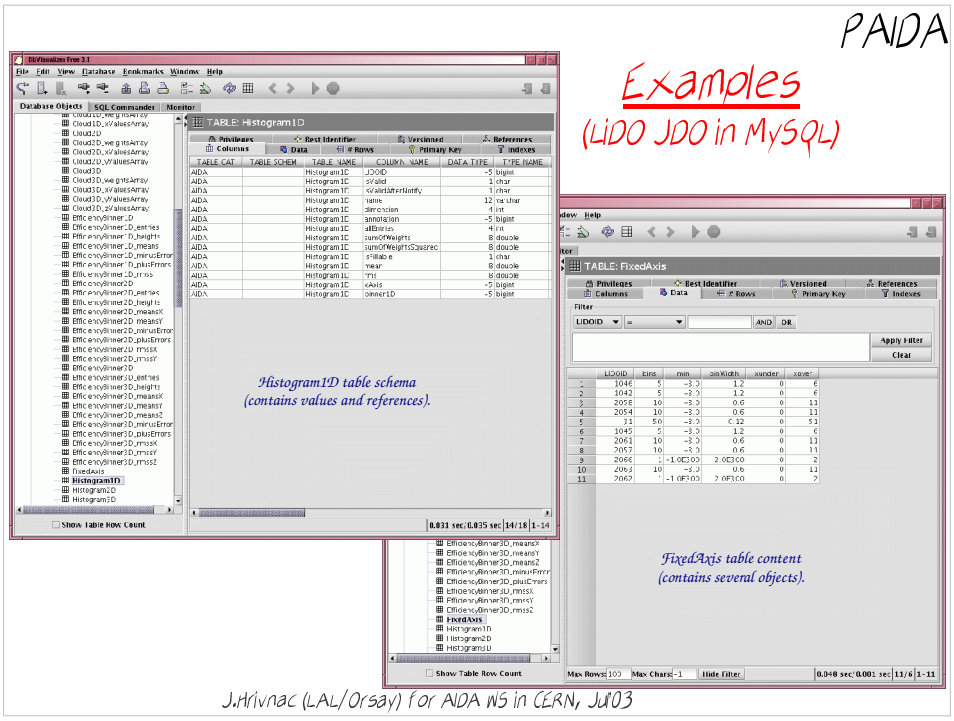
- Complete persistency for FreeHEP AIDA reference implementation have been provided (except Functions). It uses Java Data Object (JDO) standard API. It has been tested with LiDO implementation of JDO (free version of commercial tool) using several relational Databases (MySQL, McKoi, Cloudscape). Parts have been tested also with TJDO and JDORI implementation (open source).
- Data, written this way, are readable by OpenScientist and in principle by any other AIDA implementation.
- Several strategies for common approach of AIDA Persistency are proposed.



- Short overview of JDO process.
- JDO Enhancer adds persistency code into bytecode of an existing Java class. It makes this class PersistenceCapable.
- At the same time, JDO Enhancer defines a Database Schema (SQL Schema in case of RDBS) for enhanced objects. This Schema can be reused by other (non-JDO) applications.
- Classes enhanced with an enhancer of one JDO implementation are compatible with all other implementations.
- JDO Runtime (mostly via PersistenceManager) handles all persistency operations.
- All this is Java-specific. Some ideas, however, can be reused in a more general context.



- Created SQL Schema is mostly quite natural. It consists generally of tables representing PersistentCapable classes and management tables. Primitive numbers are stored directly, references are mapped to indexes.
- Created tables can be used by other applications.



- Examples of MySQL schema.
- Everything has been generated automaticaly, is manaable by JDO in FreeHEP and by plain MySQL connector in OpenScientist.

FreeHEP specific

PAIDA

Changes to FreeHEP AIDA

- Any Class can be made persistent. However:
  - Java has non-class entities:
    - Arrays: An Array is fine as long as it appears in the class interface so it can be enhanced together with the class. The problematic case is when an Array is inside a Collection. Then it can't be enhanced as a part of the Class (as it doesn't appear in the interface) and it can't be enhanced standalone, as it is not represented by a Class. A simple wrapper Class should be created.
    - Numbers: OK as they always appear in the interface (they can't be put into a Collection).
  - Only real Class (maybe abstract) can be enhanced and so referenced from another PersistenceCapable Class. Pure Interfaces should not appear as data of enhanced classes.
  - Not all system classes are PersistentCapable and system classes can't be enhanced (they come with JDK). A simple wrapper Class should be created.
- Data objects (i.e. objects to be made persistent) are not standartised in AIDA. They may (and do) differ between implementations. Can we agree on common Data objects ?

J.Hrivnac (LAL/Orsay) for AIDA WS in CERN, Jul'03

- This is mostly FreeHEP-JDO specific.
- While JDO realy can make any Class persistent, in reality there are some reastrictions. But they can be overcome (mostly by wrapping). Those problems can disappear in JDK 1.5 (thanks to templates) and JDO 2.0 (interfaces will be introduced).

PAIDA

FreeHEP AIDA in JDO
(existing FreeHEP API)

- *AidaJDOStore has the same API as AidaXMLStore. Both implement IStore FreeHEP interface.*
- *Some changes to existing implementation have been done to support JDO:*
  - *Options can be provided as PropertiesFile or Properties Object (set of name-value pairs) because real DBs have more options than XML.*
  - *IStore is decoupled from ITree. IStore can store any object, incl. ITree.*
    - *Writing: Tree can be created unmanaged and attached to IStore later.*
    - *Reading: Tree is read in from DB; it is not created and filled (as in case of AidaXMLStore).*
    - *In case of DB containing several Trees, only the first one is delivered (another options are returning a Collection of Trees or a merged Tree).*
- *Can FreeHEP IStore interface be a basic candidate for AIDA PersistenceManager ?*

J.Hrivnac (LAL/Orsay) for AIDA WS in CERN, Jul'03

- FreeHEP reference implementation has already a candidate for a Storage Manager: IStore. It is currently used for storage of AIDA Trees in XML files. It can be reused for JDO-based persistency.
- Real database technology has more options and offers richer interfaces. It could be reasonable to allow access to database features via IStore.

- JDO API has motivated sugestions for enhancements of AIDA persistency API. It is the first real database usable to persistify AIDA, everything else is just a streaming.
- Any AIDA object can be stored/read individiualy, not only within a Tree.
- Objects can be searched with a query string. The searching can be performed on the database itself, which brings big performance profit.
- All names are just suggestions (motivated by existing JDO standard).

Proposal

PAIDA

New Poposed API

(IStore)

Class IStore

```
void open();
void makePersistent(Object o);
Object getObjectId(Object o);
Object getObjectById(Object oid);
IQuery getQuery(...); // Collection c = query.execute();
close();
```

- *IStore is the main user API to AIDA persistency. It gets the properties of the storage (RW, RO, URL, user, passwd, options and hints) via Factory or Constructor.*
  - *Writing is managed by makePersistent(Object o), which just registers Object to be handled by the database. It may stored later, at te latest on the close().*
  - *Reading is managed by getObjectById(Object oid).*
  - *Searching is managed by getQuery(...). IQuery is generally constructed from filter strings. IQuery.execute() returns a Collection of results.*
- *Transaction Management is hidden behind open(), close(). It could be easily exposed (via ITransaction ?).*
- *C++ types for Object, Collection,... has to be decided.*

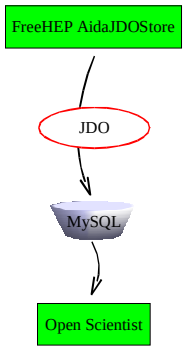
J.Hrivnac (LAL/Orsay) for AIDA WS in CERN, Jul'03

- Proposed API for IStore.
- Transaction management is hidden (behind open()/close() calls.

PAIDA

AIDA in SQL
(FreeHEP + OpenScientist)

- *In case SQL database is used, direct SQL (JDBC/ODBC) can access it.*
- *Proof Of Concept exists by creating IHistogram1D in OpenScientist from MySQL DB written by JDO from FreeHEP.*



Thanks to Guy (OpenScientist MySQL).

J.Hrivnac (LAL/Orsay) for AIDA WS in CERN, Jul'03

- Objects, stored in RDBS by JDO, are accessible from other AIDA implementations. Proof-of-concept has been provided with Open Scientist reading Histogram1D from MySQL database written by JDO-enhanced FreeHEP.
- It is important to agree on common interfaces so that applications can realy cooperate. More about this below.

Proposal

PAIDA

Common SQL tables

- *Common API can be defined on three levels:*
  1. *All implementations use the same Schema of SQL tables.*
     - ☺ *All implementations can directly read all data. High level of interoperability.*
     - ☹ *AIDA data objects (Java, C++) implementation should be agreed (otherwise we will suffer from conversion overhead).*
     - ☹ *No profit from special performance of SQL Databases (i.e. from processing queries inside a database).*

J.Hřivnač (LAL/Orsay) for AIDA WS in CERN, Jul'03

- Common storage mechanism can be agreed on top of common storage API (IStore). It can be done on three levels.
- We can use exactly the same SQL Schema (same tables).

Proposal

PAIDA

Standard SQL scripts

- *Common API can be defined on three levels:*
  - *Each SQL database will contain tables with SQL scripts to create/write/read/search AIDA objects.*
    - *Profits from native performance of SQL databases.*
    - *Quite clean interoperability.*

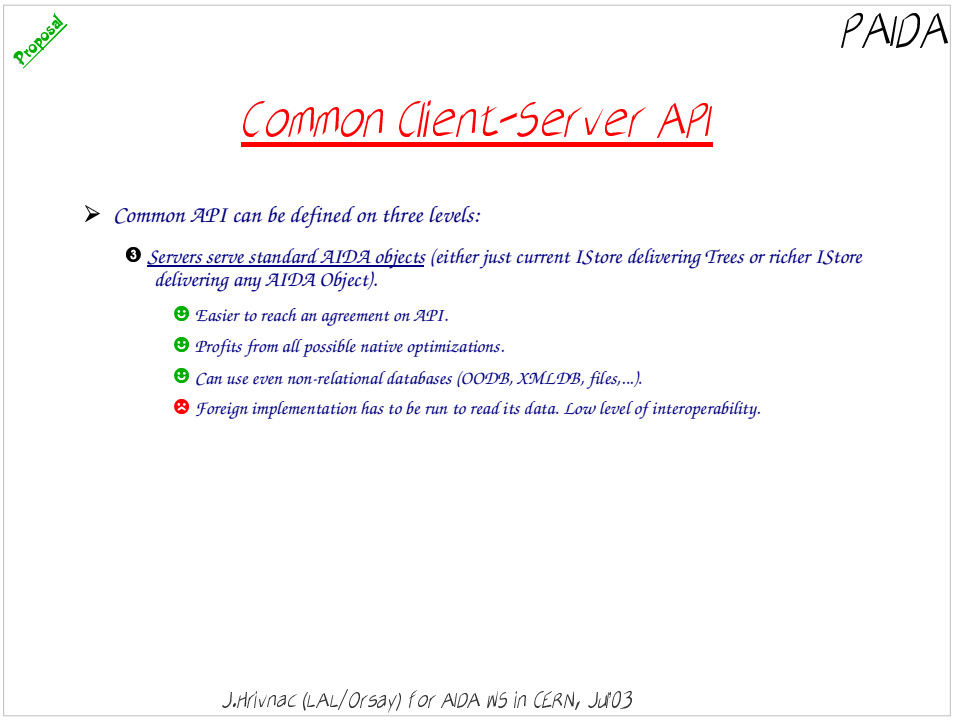
Read Table	
object	script
Histogram1D_attributes	"select mean,rms,... from Histogram1D"
Histogram1D_content	"select ..."

Write Table	
object	script

Create Table	
object	script

J.Hřivnač (LAL/Orsay) for AIDA WS in CERN, Jul'03

- We can store SQL scripts for accessing database in the database itself. It will allow us more flexibility.



- We can work on the Client-Server level, leaving all persistency management on the implementations.

PAIDA

Summary

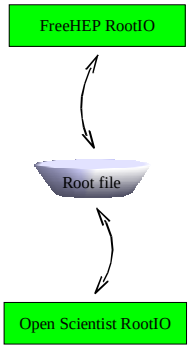
- *Proposals for two new APIs:*
  - *Persistency API (User): IStore, IQuery.*
  - *Storage Access API (Developper) at the level of:*
    - *Common SQL tables*
    - *Standard SQL scripts*
    - *Client-Server API*
- *Proof-of-concept done for FreeHEP & OpenScientist.*
- *Works in (both) languages, C++ functionality is (innevitably) limited:*
  - *In Java environment:*
    - *can profit from full database functionality (caching, transactions, lazy loading/updating,...),*
    - *can use any database (SQL, OO, files,...).*
  - *In C++ environment:*
    - *can read/write/search,*
    - *can use SQL database.*

- Two new proposals motivated by mature JDO standard.
- Java envionment is more functional(as usually).

PAIDA

And what about Root files ?

- *Object interchange using Root files works between FreeHEP and OpenScientist (both ways).*
- *Details have to be clarified (to write TObjects (less fuctional) or AIDA Objects (not readable by Root),...).*
- *IDO interface under consideration.*



Thanks to Tony (FreeHEP RootI, Peter (FreeHEP RootO), Guy (OpenScientist RootIO).

J.Hrivnac (LAL/Orsay) for AIDA WS in CERN, Jul'03

- Both FreeHEP and Open Scientist can read/write Root files.