Time and storage patterns in Conditions: old extensions and new proposals

- The “Extended” Conditions Interface (MySQL)
- The ATLAS Experience
- Dealing with files or references
- Why we need a new interface
- Features being investigated
- Common points and diferent aproaches
- Work Plan Proposal
- The Interface Specification Proposal

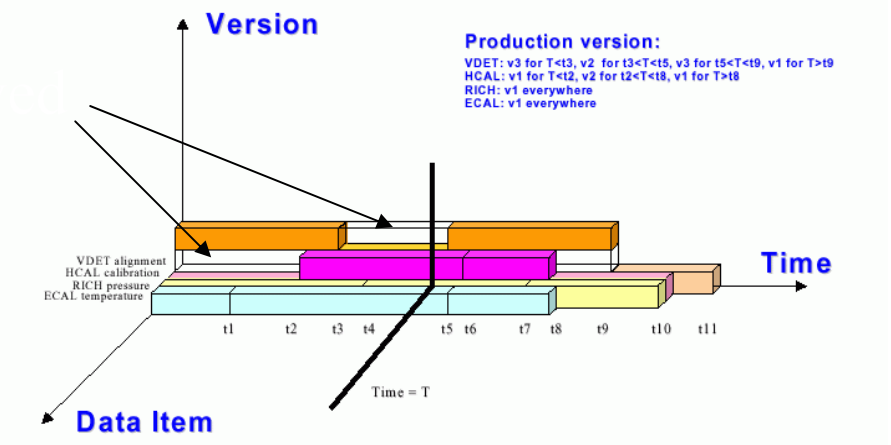
The Extended Conditions Interface (MySQL)

- The main ATLAS domains with time management:
  - Calibration/Alignment ; (Slow) Control; Configuration; Monitoring
- CondDB initially developed by BaBar using an ODBMS.
- It was re-designed at CERN and later re-implemented in ORACLE
- We implemented in MySQL and saw the need to extend:
  - It contained only BLOBS with time intervals, versions and tags.
  - outside of the “ATLAS Rec. Framework” the objects were meaningless
  - The time behavior was not appropriate for Control, Configuration online
  - It did not scale with data that keeps being loaded.

The Extended Conditions Interface

- Schema in DB → The CondDBtable container:

  A single transient C++ class for a generalized table, including arrays of any types as cells.
- Open Source RDBMS → available on all OS+Comp.
- Improved folders

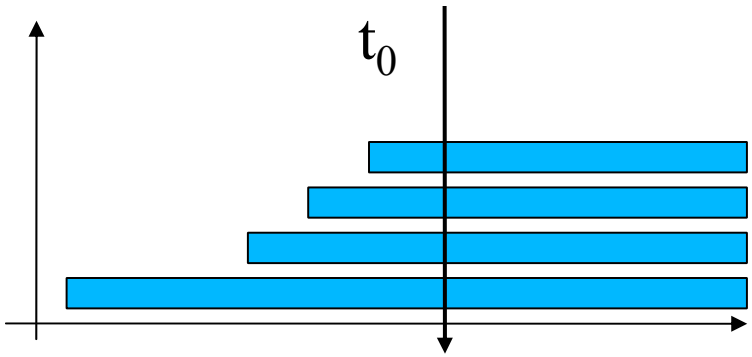


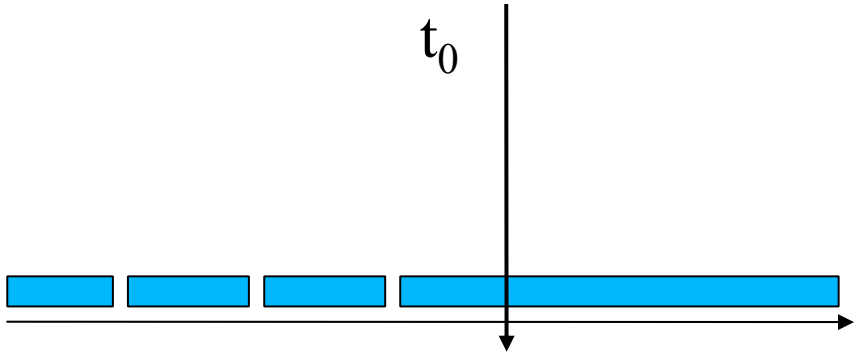
- Online Folders: not knowing the end validity

Generalized Container

- Abstract Interface -> ICondDBTable
- Performance Optimized implementation based on
  - Variable type STL containers
  - Numerical data in binary form
  - Including vectors as DB entry values
- Used in all DB operations
- Being extended for generic object columns
- Extend to ROOT or POOL container behavior?

Online Folders





- Normal folder
  - Diff versions for t_0
  - Many versions for large t
  - Can correct for old times
  - Versions can be tagged
  - Can be our containers or blobs
- Online Folder
  - Any update cuts the previous interval
  - Single version
  - Can not correct
  - Fast and efficient

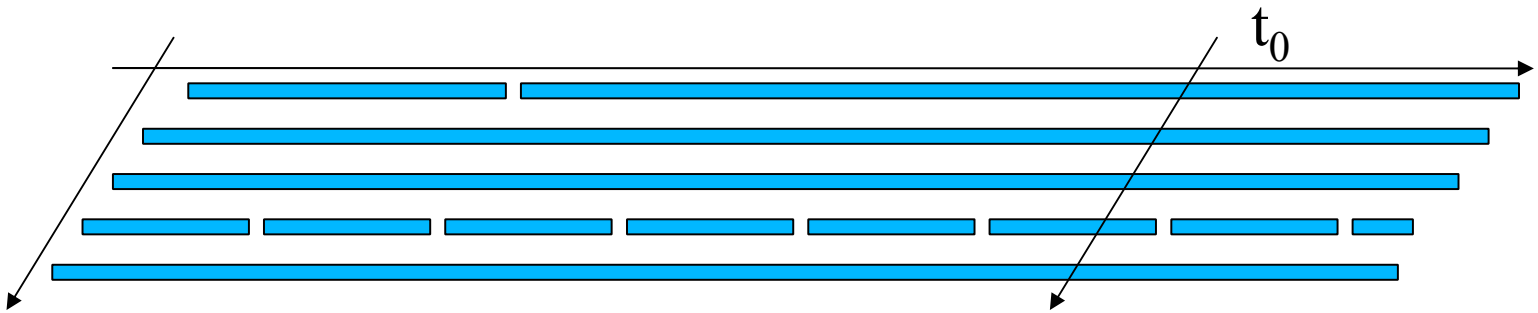
Calibration/Alignment

DCS

DAQ

LOG

Asynchronous Folders (ID)



- DCS one channel varies and the others do not
- Configuration/geometry/parameters
  - A small number of parameters is changed often
  - The others are kept valid

Often used for DCS, configuration, detector description

Partitions on time (scaling)

- To scale the DB servers the DB administrator can partition the folders in time such that ex:
  - year 2004 in server atl04 DB conditions04
  - year 2005 (up to June) in server atl05 DB conditions05-1
  - year 2005 (after June) is in server atl05 DB conditions05-2
  - ...
- This is transparent to the user since the first query is always to the partition master
- Objects overlapping the partition boundaries are replicated in both partitions.

Running for ATLAS

- DCS – Detector (Slow) Control
  - PVSS (SCADA) -> Conditions
- T/DAQ – CDI (Interface)
  - Online Information System, Messages -> Conditions
- Reconstruction Framework (ATHENA)
  - Conditions <-> Athena (Conversion Service)
- ATLAS Sub-detector configuration and monitoring

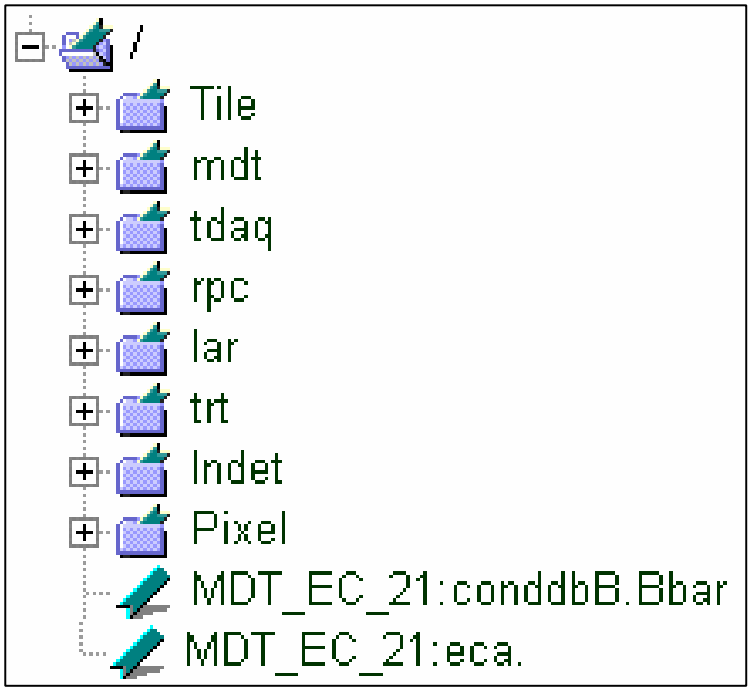
PVSS Manager

CDI Interface

DB Container
Conversion Service

CondDB Browser

- C++ API -> php binding -> Browser
- Integrated with the ATLAS NOVA database system



Can see the data in the Gen. Container

Found 10167 row(s).

1 2 3 4 5 6 7 8 9 10 next

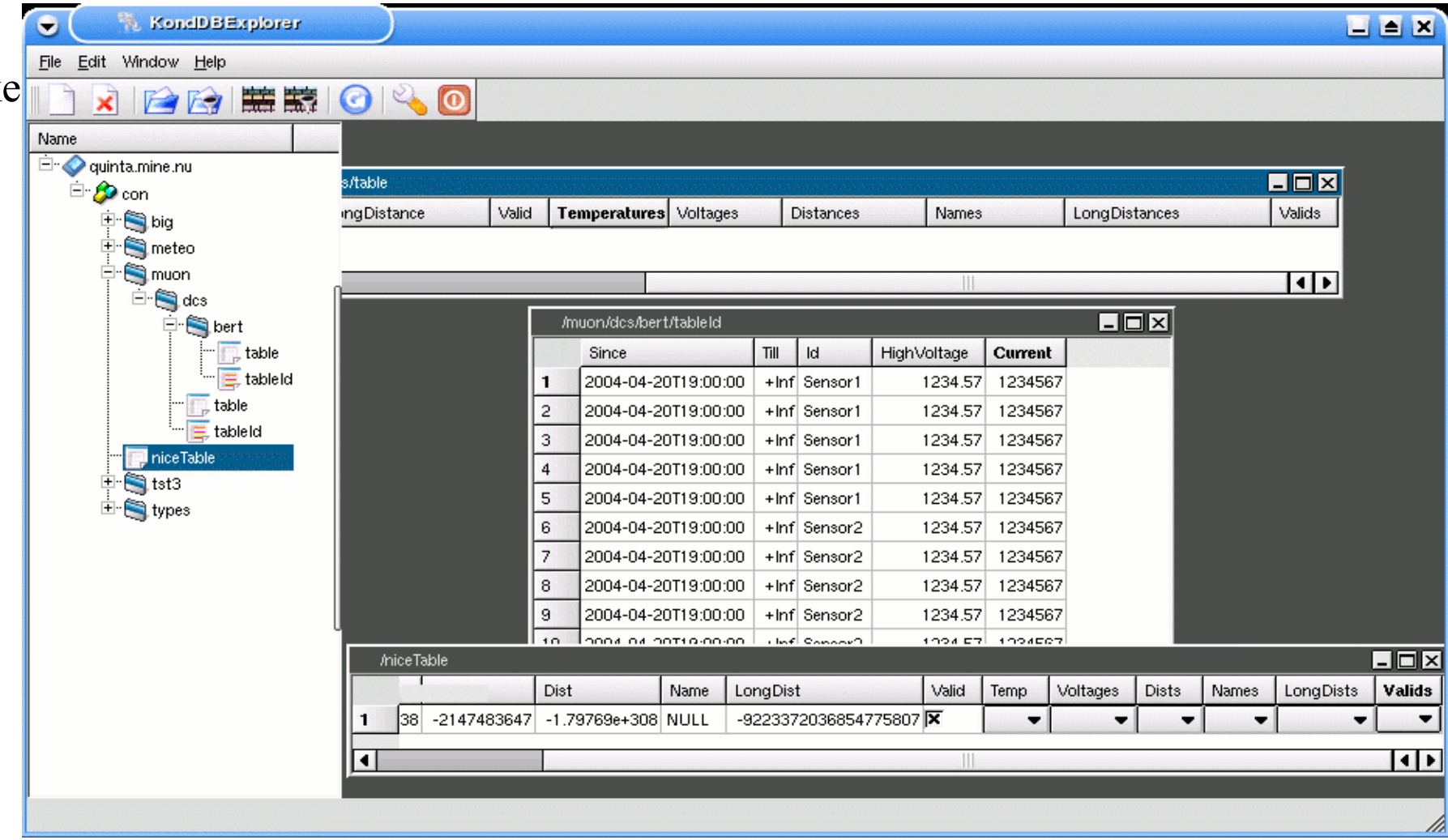
	Since (GMT):	Till (GMT):	chamber	seqnr	temps
[0]	-Inf	1970-Jan-1 0:0:0.0	NULL	NULL	NULL
[1]	2004-Jun-7 10:32:34.289000000	1970-Jan-1 0:0:0.0	BIL1	1929	21.784
[2]	2004-Jun-7 10:32:35.320000000	1970-Jan-1 0:0:0.0	BIL2	1929	21.956
[3]	2004-Jun-7 10:32:36.351000000	1970-Jan-1 0:0:0.0	BML1	1929	21.963
[4]	2004-Jun-7 10:32:37.382000000	1970-Jan-1 0:0:0.0	BML2	1929	
[5]	2004-Jun-7 10:32:38.414000000	1970-Jan-1 0:0:0.0	BOL1	1929	
[6]	2004-Jun-7 10:32:39.461000000	1970-Jan-1 0:0:0.0	BOL2	1929	
[7]	2004-Jun-7 10:32:40.492000000	1970-Jan-1 0:0:0.0	EOL	1929	
[8]	2004-Jun-7 10:32:41.539000000	1970-Jan-1 0:0:0.0	EOS	1929	
[9]	2004-Jun-7 10:32:42.586000000	1970-Jan-1 0:0:0.0	EIL	1929	
[10]	2004-Jun-7 10:32:43.648000000	1970-Jan-1 0:0:0.0	EIS	1929	
[11]	2004-Jun-7 10:32:44.695000000	1970-Jan-1 0:0:0.0	EML	1929	
[12]	2004-Jun-7 10:32:45.742000000	1970-Jan-1 0:0:0.0	EMS	1929	

21.963
0
21.67
0
22.338
0
21.755
21.963
21.735
21.995
0
0
0
0

The new kde based ConDB editor

Recent development by João Simões.
Try it on afs: ~aamorim/public/konddbexplorer

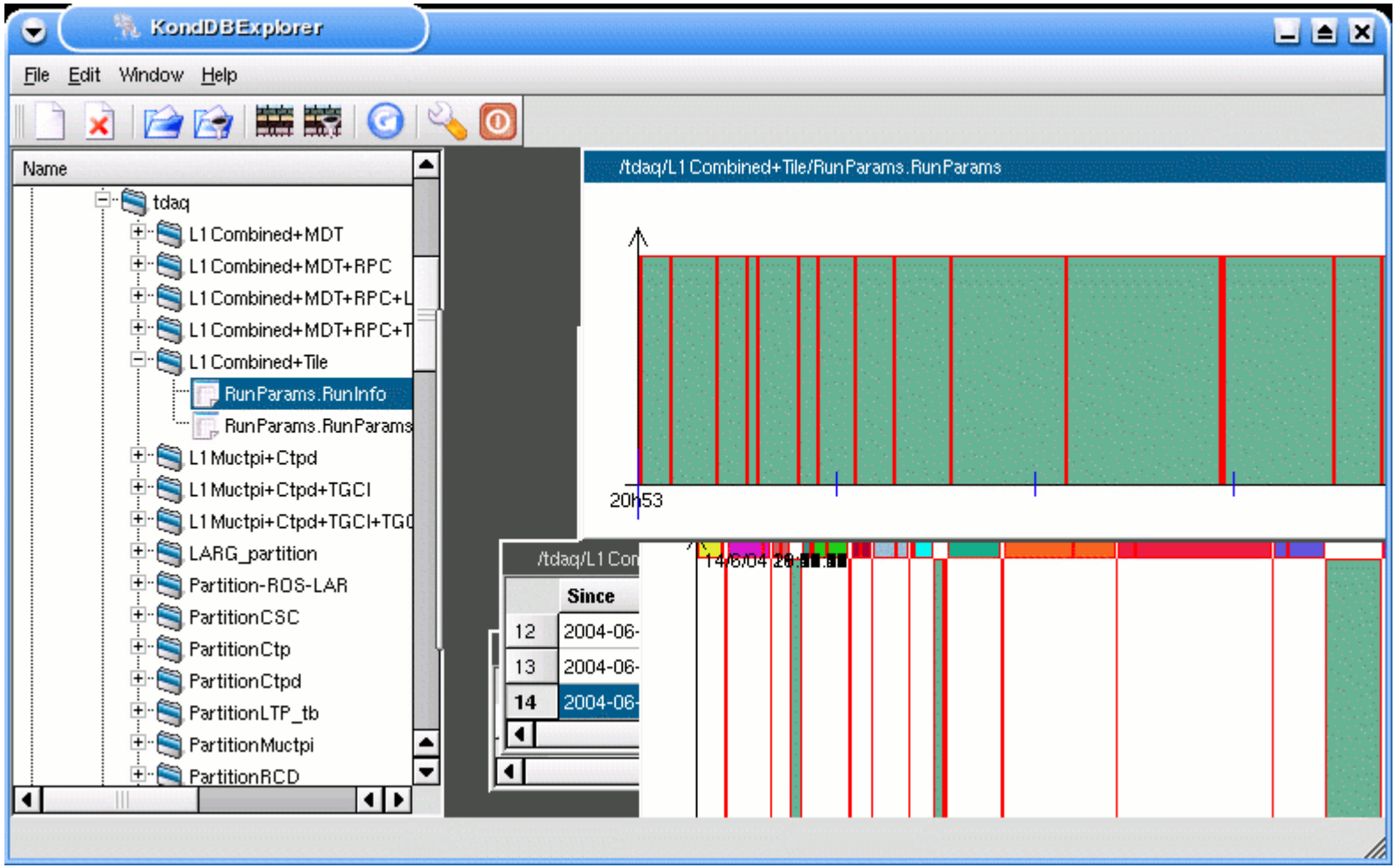
Installation configure/make
Being tested.



kconddbexplorer

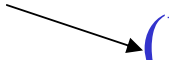
USE: cache on

Host: atlobk02.cern.ch
Database: conditions_ctb_2004
User: conditions



ATLAS Test-beam operation

- Combined test-beam (ATLAS slice test)
- Online MySQL server -> Offline MySQL server
- ~ 8.5 GBytes of database data
- 1859 Folders in total
- 43 Generic Containers; 1809 Online folders; 7 Online Asynch.



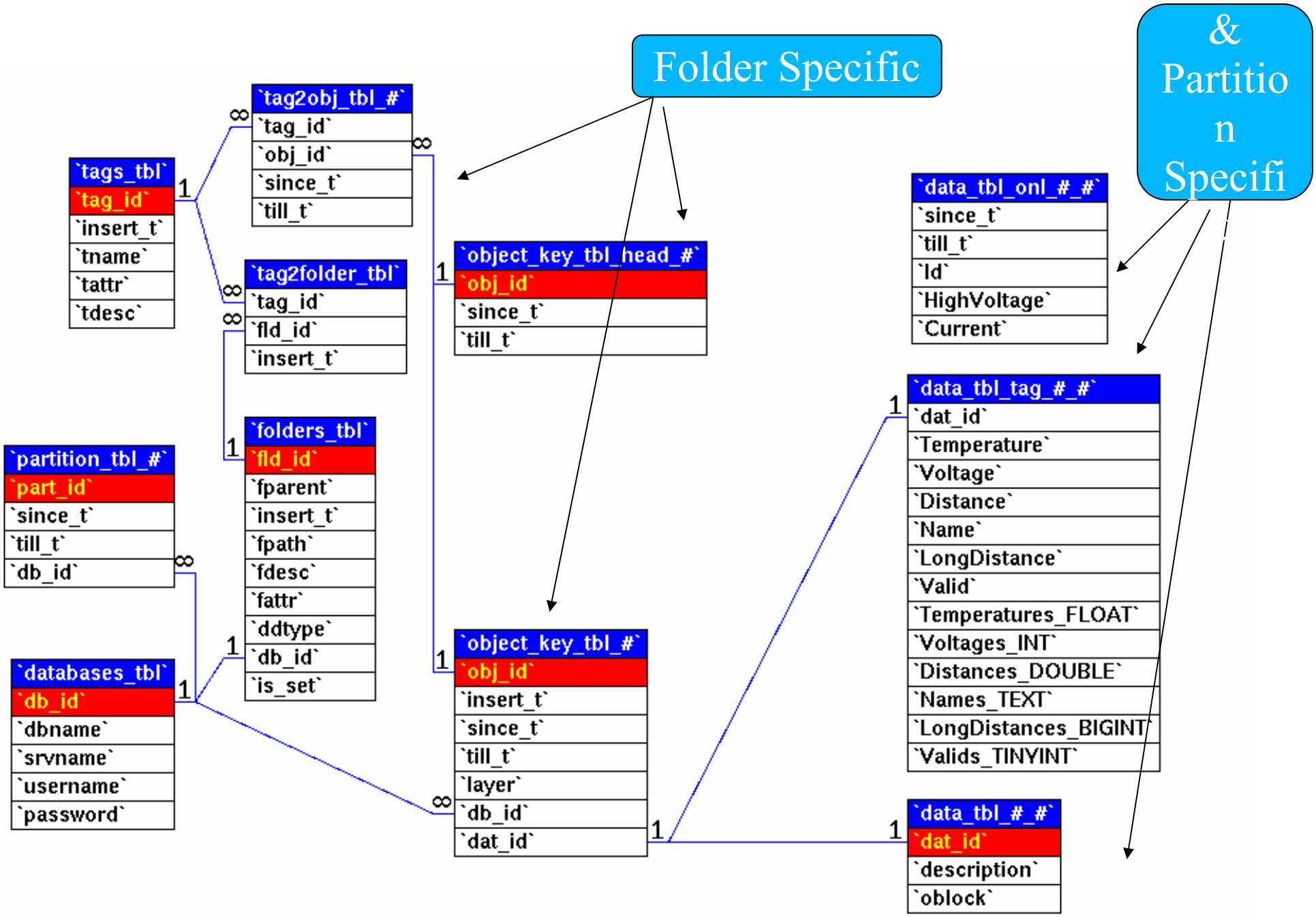
(References to NOVA tables)

More detains in the page:
http://www.abstracture.de/atlas/ctb.html

Running successfully!

Simulation/Reconstruction: Calibration and Alignment is a refinement – Not yet tested extensively

Schema - Let the DB do its work

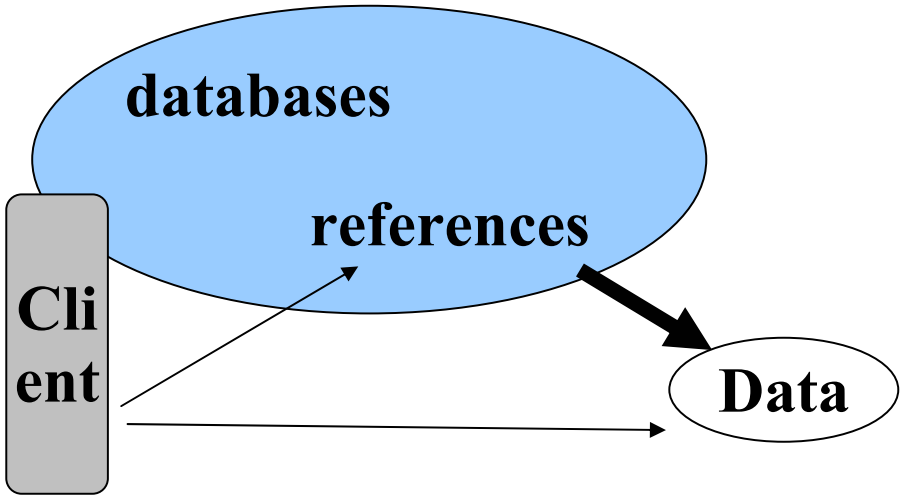


Could it all be files?

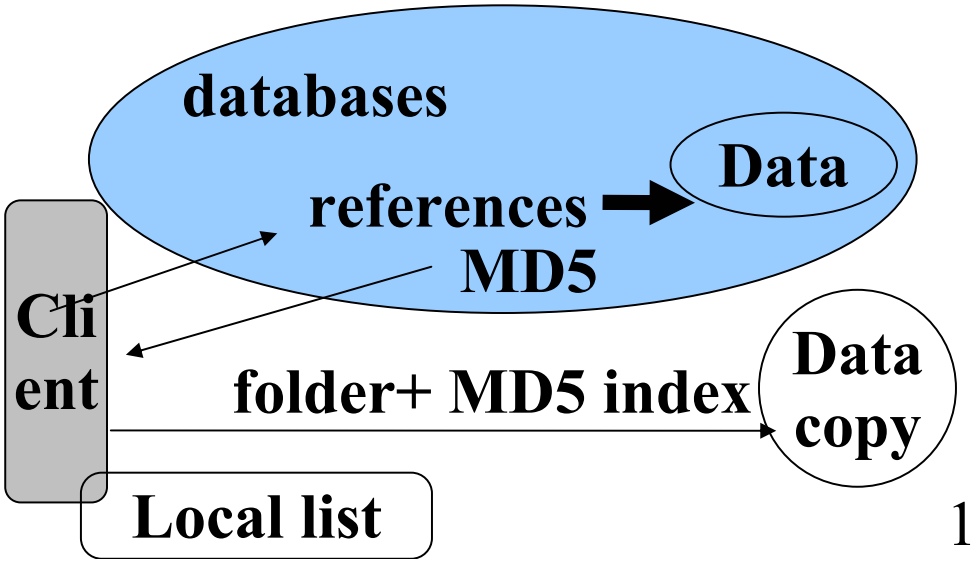
- Online:
  - Many small objects with irregular access pattern
  - Distributed environment: all files should be accessible always to all nodes.
  - Event -> Index -> (logical) Calibration Files -> Files
- Offline
  - Data Sets associated with Calibration/Online files in Index File -> (logical) Calibration Files -> Files
  - All (small) files should be accessible always
- DB effort shifted -> replica catalog and to file server.
- Bringing to the client objects (file) not needed for the job.

Dealing with references

- References to external objects have to be managed very carefully
- Any reference is a possible break of integrity: “referential integrity” (even using logical file names)
- A reference to an object in a file in my laptop can be lost just because it is stolen!
- Instead of:



Investigate:



Why we need a new interface

- The CondDB user interface was not intuitive
- With our extensions were had to fit in the interface
- The DB must provide centralized object description storage for all objects associated with the external technologies: POOL, ROOT
- CondDBTable needs to be extended and revised
- These developments are independent of MySQL and should be implemented both in MySQL and ORACLE
- For “folders with version mechanism” was improved but still has problems for growing number of updates.
- Tagging is a mess

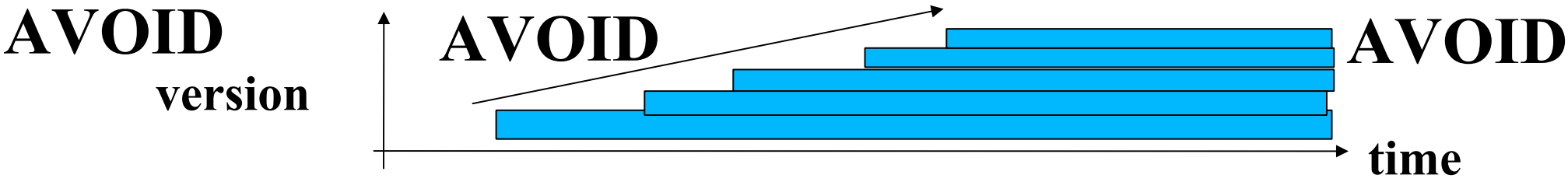
Features Being Investigated

- Abstract interface: implementations in ORACLE and MYSQL
  - interface classes associated to specific objects ex: FolderManager object is one particular folder
  - Possible namespace
  - Use exceptions plus a wrapper to access without exceptions
- Keep Hierarchical view (folders and folder sets (/../../../..))
- Extended collection view (generalized CondDBTable)
  - Column types (simple, var-array, extended objects)
  - A POOL or ROOT class as a column type!
  - also column with variable type (from NOVA functionality)
  - line restriction and column projection by the server on query

Extended Time behaviour

1) Online folder type: no versions, t-interval:
   Cut at insertion time t_2 in $[t_1,inf[$ -> $[t_1,t_2]+[t_2,inf[$
   a) Single Object as a function of time
   b) Collection with "id" s that evolve differently

2) Offline folder including versions and tags.
   Usual version time diagram with:
   null object suppression +
   cut at insertion time for t > last start time in folder (v=0)



3) Data mining folder as in 2) but with internal time intervals per line. ex: Temperatures for 1 month.

4) Bare – time independent nature

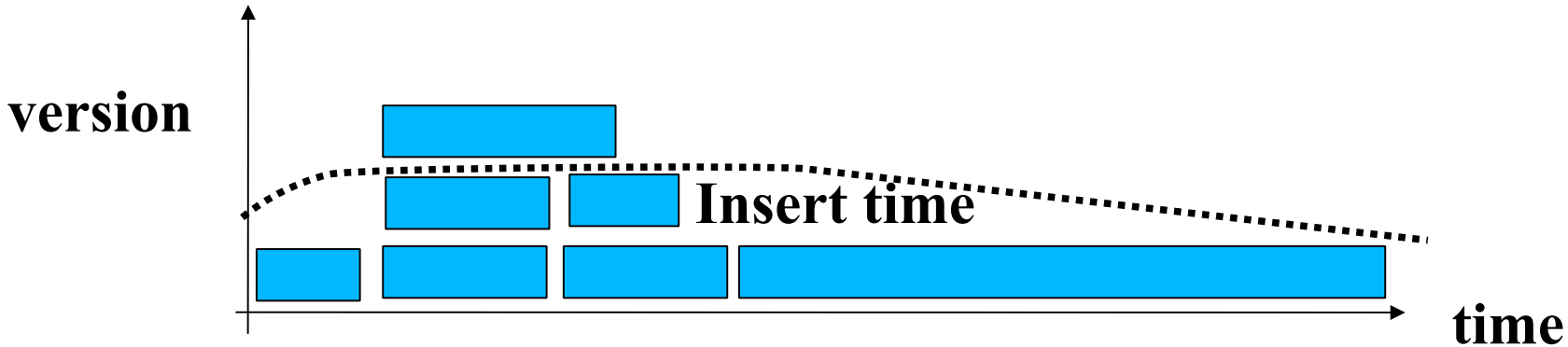
Investigations on Tagging

1) Tag from head

2) Tag from tag: create a tag with the objects of another tag

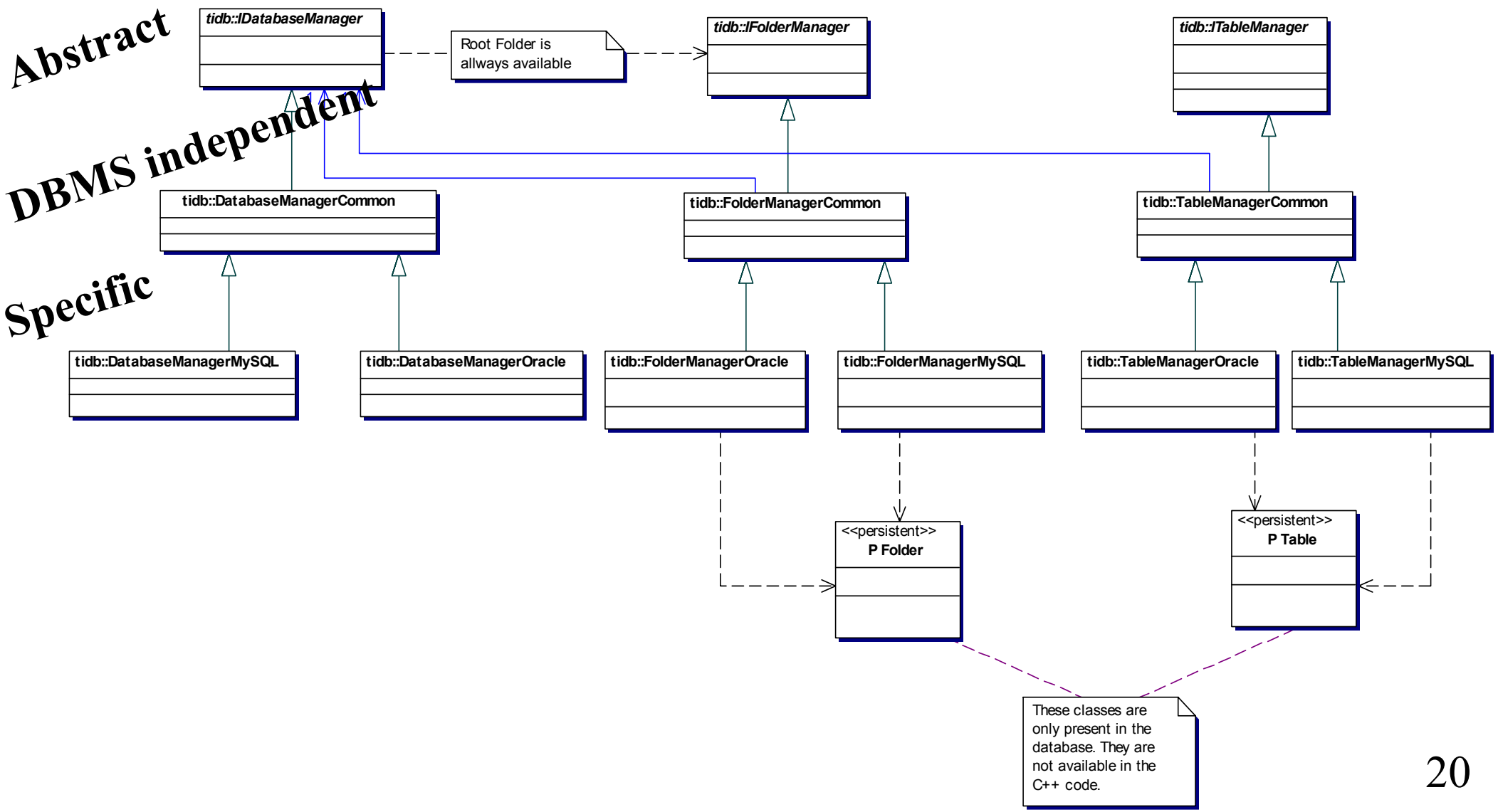
3) Tag to tag: hierarchical tags that point to several other tags

4) Create tag or Re-tag to the “old” head with insertion time less or equal to a given time (from BaBar)



5) Use a re-tag time interval that only changes objects that are contained in a user time interval.

Redesign the Interface and Investigate a three layer approach



Common points

- Folder specific tables
- Table data with relational nature instead of single blobs
- Storage of objects as blobs in cells
- Abstract interface
- MySQL and ORACLE implementations

Different Aproaches

- Focus on interface modifications - *schema*
- Exploit the ORACLE and MySQL optimization separately - *common RAL*
- *Keep "online" folders and "offline" folders – common*
- *Store Object descriptions for each "column" in a database facility – leave this to the applications*
- *Emphasis on storing data – emphasis on storing references*
- *Exploit partioning to solve scaling problems – keep data payload out*

Interface Modifications

- Old interface:
  - All managers except IconDBManager were statless
- New interface:
  - All managers are associated with the “container” ones
- Advantages:
  - Simpler: less parameters in the methods
  - Faster: All processing of parameters (involving db) does not have to be repeated.

ORACLE MySQL optimization

- In ORACLE:
  - Use ORACLE “dblinks” to access different databases in partitioning
  - use stored procedures for folder/table creation
  - use array management
- In MySQL
  - store object type “id” in appended to column name
  - ...
- RAL – avoid ODBC in MySQL ...

Data Container:

- Generalized DB Table -> Mem. Data Container or
  -> Relational Ttree
  -> Relatiional POOL

- The package can still be used standalone but the user can see the DB objects in the manner that is more friendly to him.

Work plan proposal

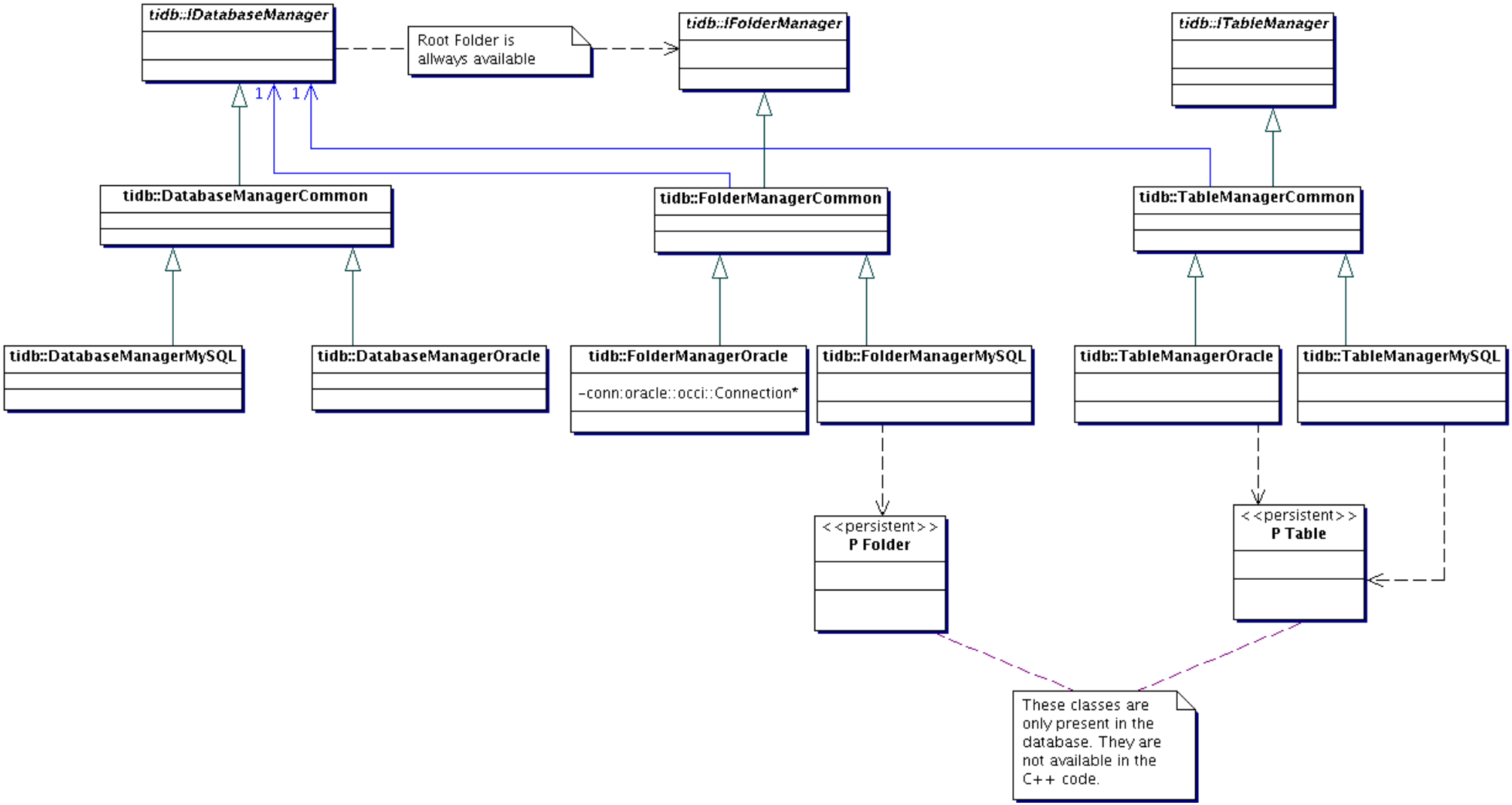
- Converge on the interface soon
- Have the implementation of the object type storage as optional.
- Converge or partitioning or make it optional
- Have 3 implementations based on:
  - MySQL – mainly us (help usefull in 1 or 2 months)
  - ORACLE – mainly us (help usefull in 1 or 2 months)
  - RAL - mainly CERN/IT (...)
- Evolve the container Object minimal interface such that is can be covered by (db),(ROOT/Tree),(POOL)

The Interface Specification Proposal

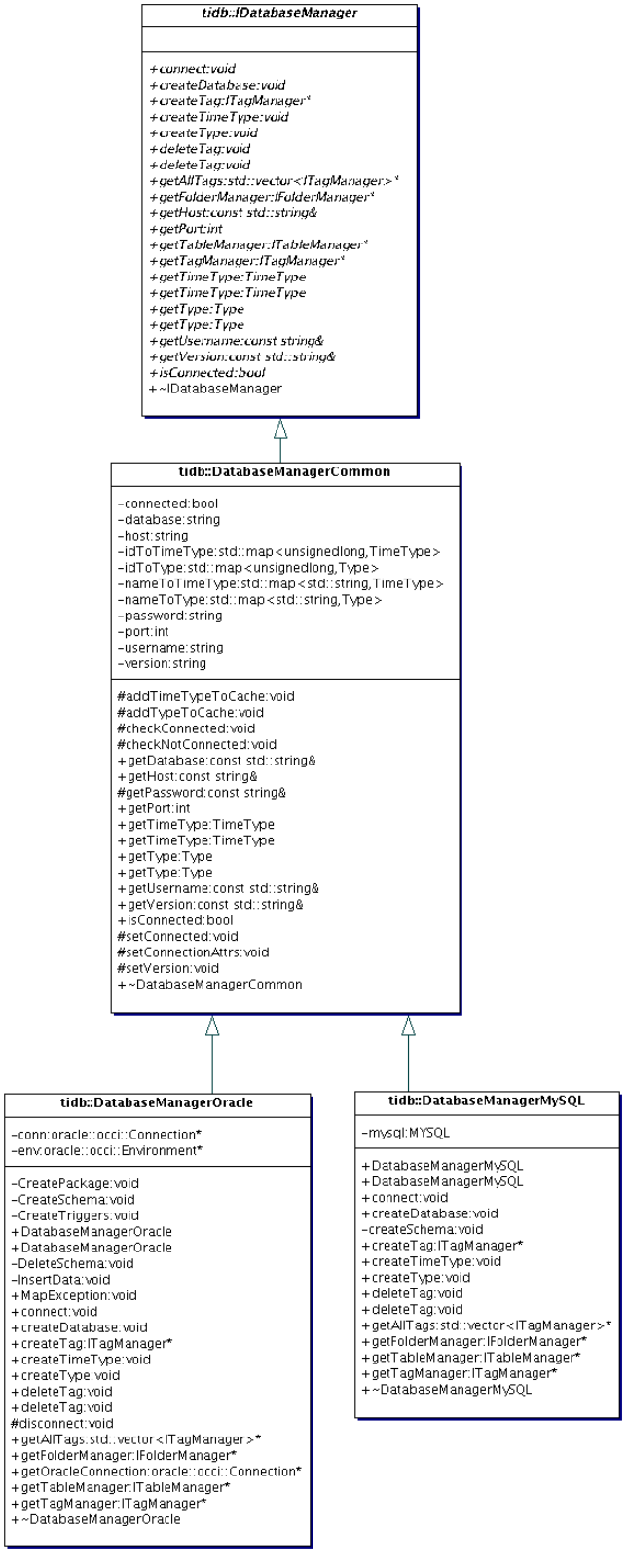
Consider TIDB can be placeholder

“Time oriented instrumental databases”

Databases, Folders and Tables

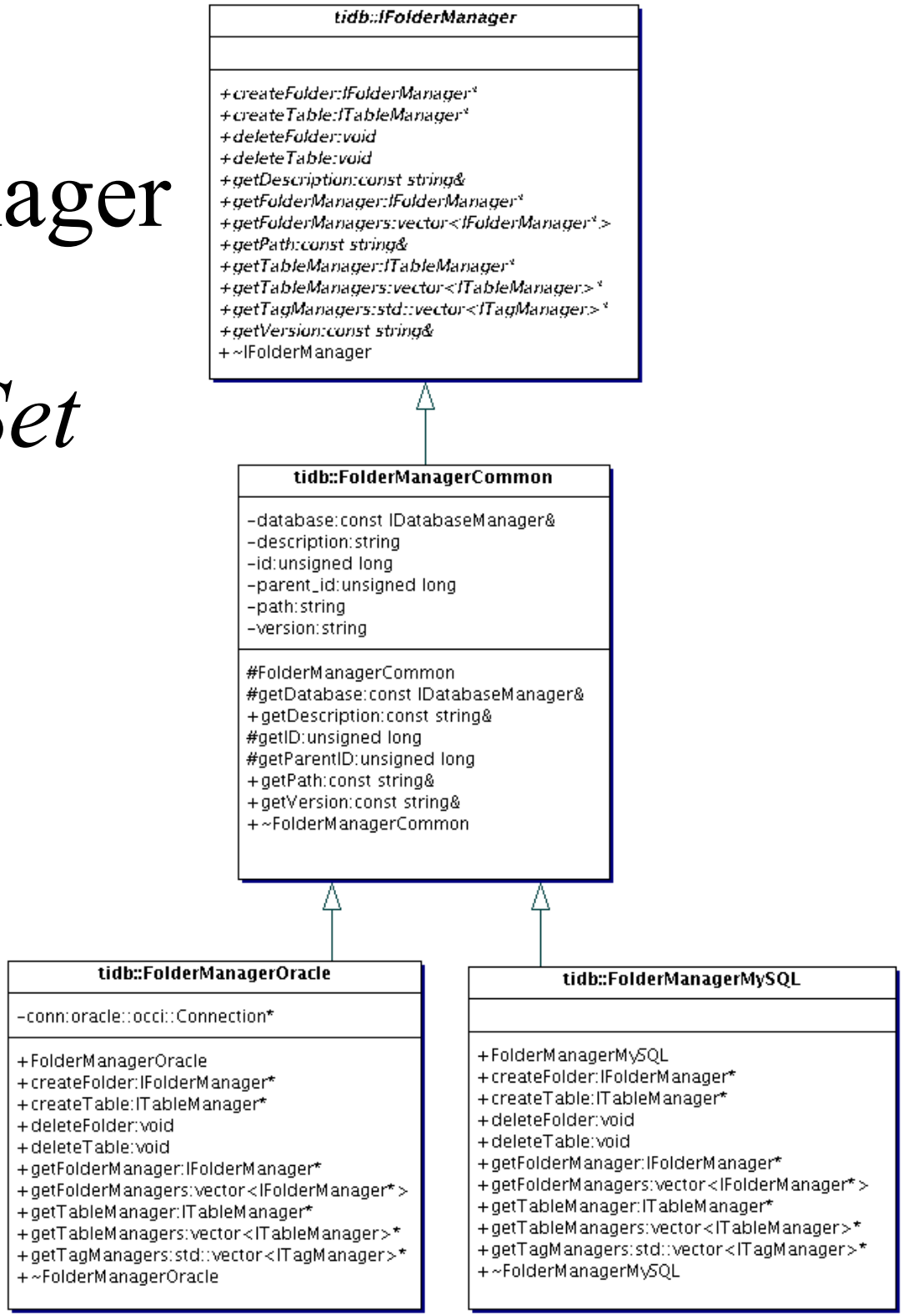


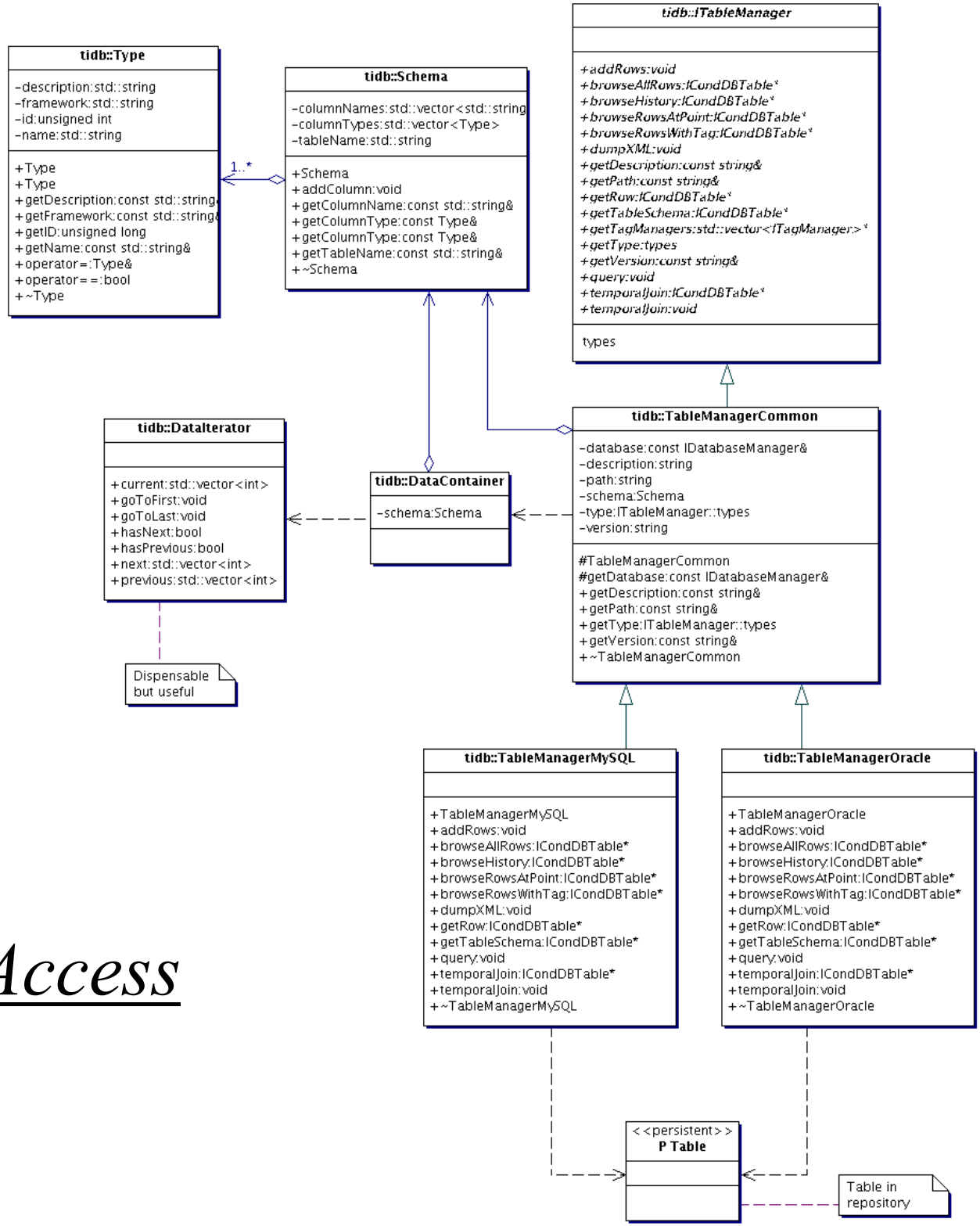
Database Manager



Folder Manager

old FolderSet concept

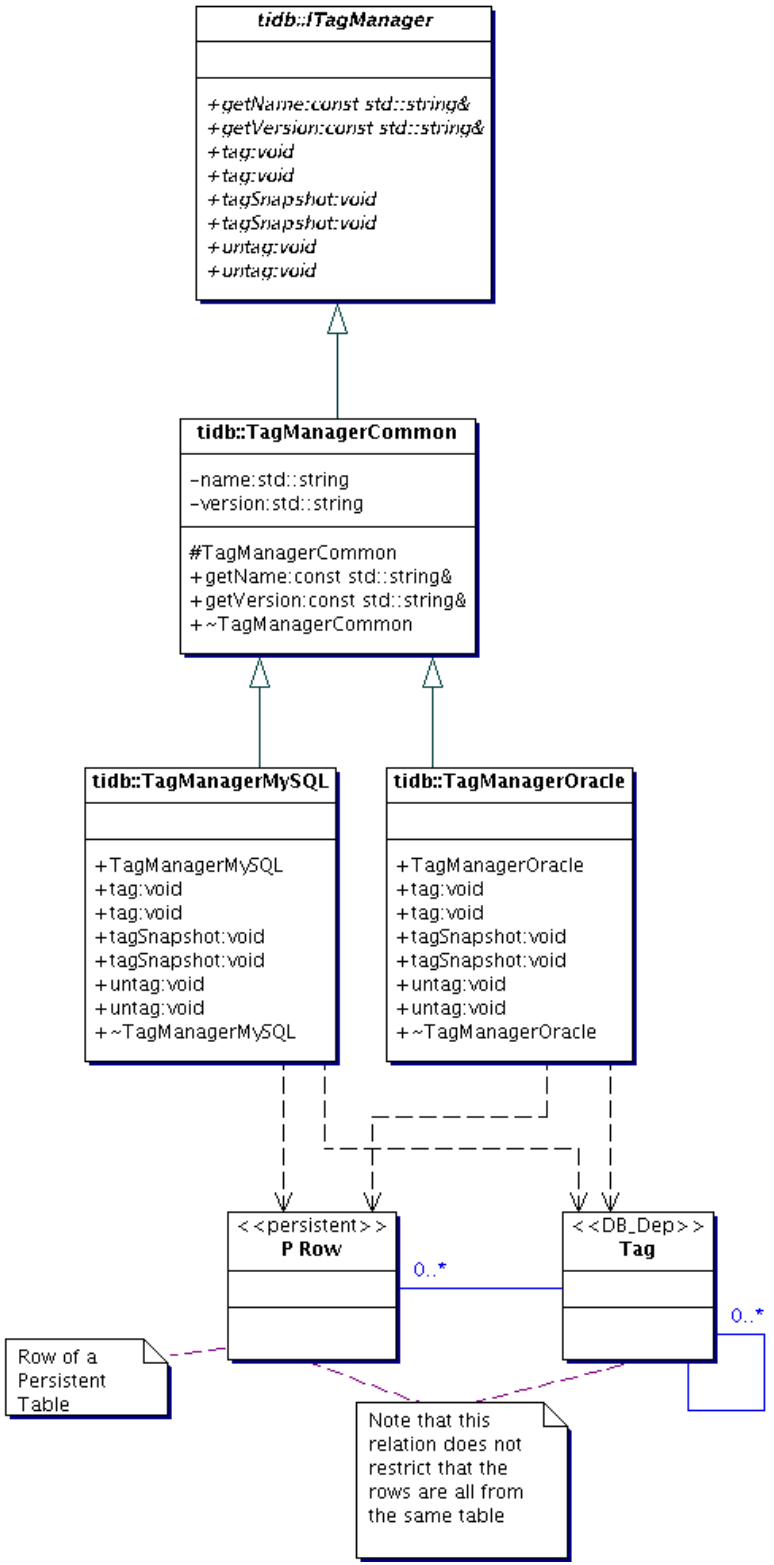




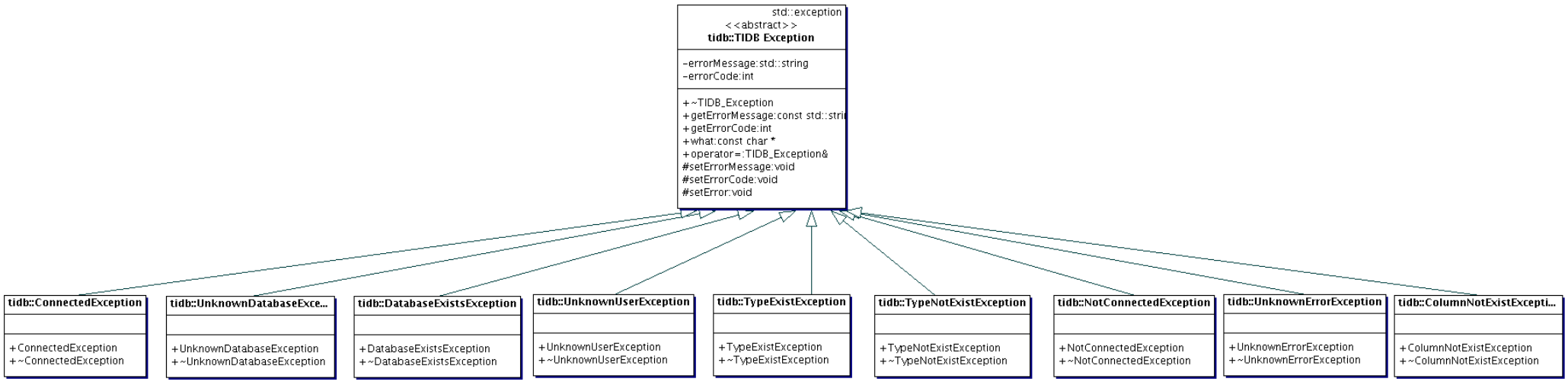
TableManager

old Folder and DataAccess

Tag manager


tidb::ITagManager
+getName:const std::string&
+getVersion:const std::string&
+tag:void
+tag:void
+tagSnapshot:void
+tagSnapshot:void
+untag:void
+untag:void
tidb::TagManagerCommon
-name:std::string
-version:std::string
#TagManagerCommon
+getName:const std::string&
+getVersion:const std::string&
+~TagManagerCommon
tidb::TagManagerMySQL
+TagManagerMySQL
+tag:void
+tag:void
+tagSnapshot:void
+tagSnapshot:void
+untag:void
+untag:void
+~TagManagerMySQL
tidb::TagManagerOracle
+TagManagerOracle
+tag:void
+tag:void
+tagSnapshot:void
+tagSnapshot:void
+untag:void
+untag:void
+~TagManagerOracle
<<persistent>>
P Row
<<DB_Dep>>
Tag
0..*
0..*
Row of a Persistent Table
Note that this relation does not restrict that the rows are all from the same table

Exceptions


std::exception
<<abstract>>
tidb::TIDB Exception
-errorMessage:std::string
-errorCode:int
+~TIDB_Exception
+getErrorMessage:const std::stri
+getErrorCode:int
+what:const char *
+operator=:TIDB_Exception&
#setErrorMessage:void
#setErrorCode:void
#setError:void
tidb::ConnectedException
+ConnectedException
+~ConnectedException
tidb::UnknownDatabaseExce...
+UnknownDatabaseException
+~UnknownDatabaseException
tidb::DatabaseExistsException
+DatabaseExistsException
+~DatabaseExistsException
tidb::UnknownUserException
+UnknownUserException
+~UnknownUserException
tidb::TypeExistException
+TypeExistException
+~TypeExistException
tidb::TypeNotExistException
+TypeNotExistException
+~TypeNotExistException
tidb::NotConnectedException
+NotConnectedException
+~NotConnectedException
tidb::UnknownErrorException
+UnknownErrorException
+~UnknownErrorException
tidb::ColumnNotExistExcepti...
+ColumnNotExistException
+~ColumnNotExistException