TTree / SQL

Philippe Canal (FNAL)
2005 Root Workshop

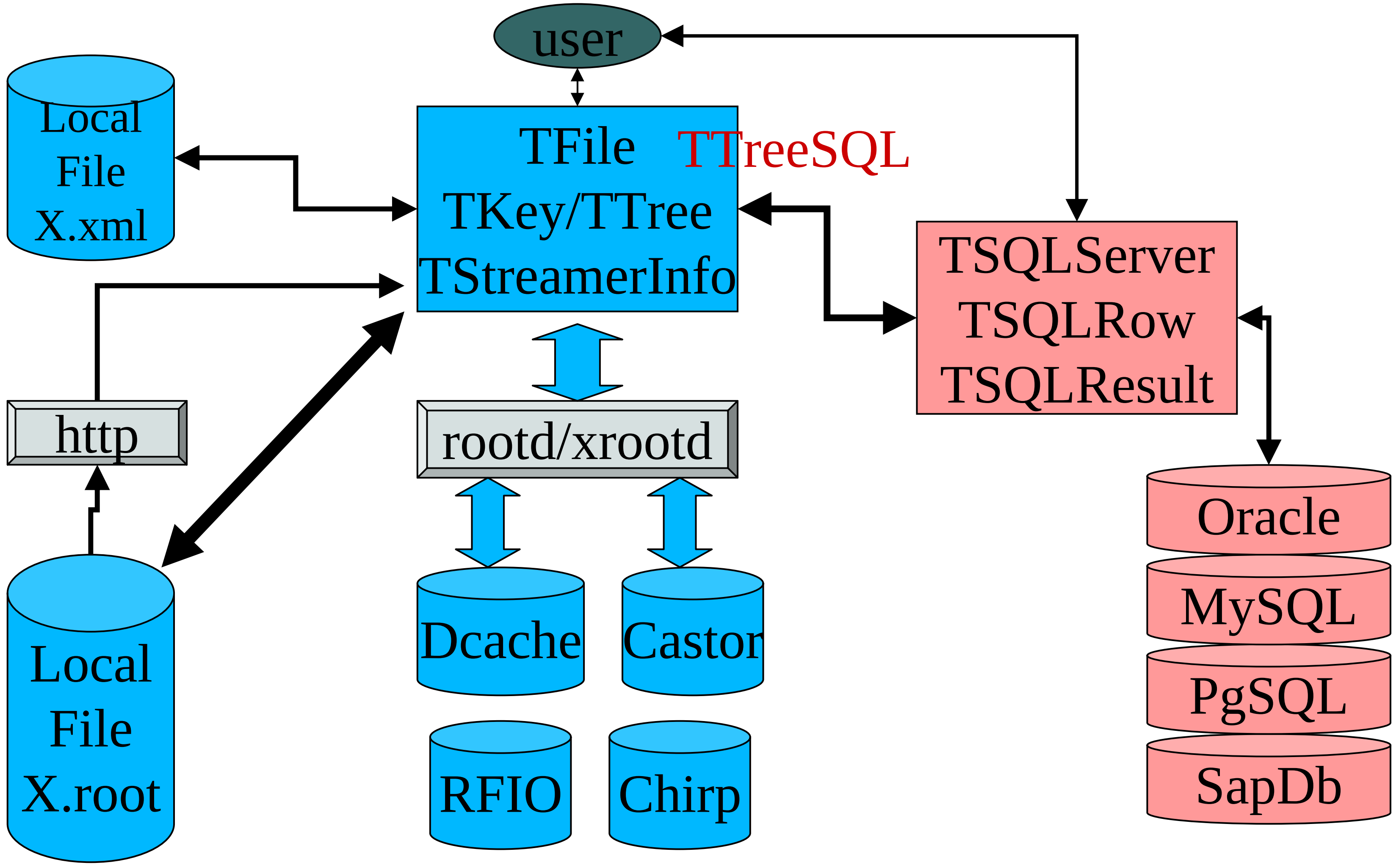
New RDBMS interface: Goals

- Access any RDBMS tables from TTree::Draw

- Create a TTree in split mode
  - ➔ creating a RDBMS table and filling it.
- The table can be processed by SQL directly.
- The interface uses the normal I/O engine
  - including support for Automatic Schema Evolution.

- Convergence between ***RAL*** interface and the TSQL interfaces

File types & Access in 5.04/xx



TTree with SQL database back-end

- Uploaded in CVS repository of first version of TTreeSQL
  - support the TTree containing branches created using a leaf list (eg. hsimple.C).

```
ntuple->Branch("main",&mytest,"px/D:py/F:pz:random:i/I:c/B");
ntuple->Branch("string",(void*)str,"str/C");
```

- Add an interface to read the proper TTree object depending on the backend
  - Something like TTree::Open using the Plugin Manager
- Extend TTreeSQL to support TBranchElement
- Implement proper schema evolution support
  - The main design problem is how to save/retrieve the TProcessID/TStreamerInfo.
  - One possibility is to use the same mechanism currently in use in TXMLFile

TTreeSQL Syntax

- Currently:
  - ROOT:

```
TFile *file = new TFile("simple.root","RECREATE");
TTree *tree; file->GetObject(“ntuple”,tree);
```

  - MySQL:

```
TSQLServer*dbserver = TSQLServer::Connect("mysql://…”,db,user,passwd);
TTree *tree = new TTreeSQL(dbserver,"rootDev","ntuple");
```

- Coming up:

```
TTree *tree = TTree::Open(“root:/simple.root/ntuple”);
```

```
TTree *tree = TTree::Open(“mysql://host../rootDev/ntuple”);
```

Support for TBranchElement

- Will add the creation of auxiliary tables
  - table of TStreamerInfos
- Will add support for ‘blob’ data field to support unsplit object.
- Will need support for ‘collection’
  - either by using additional ‘linked’ tables
  - either by using ‘blob’ data field

TTreeSQL Optimization

- On a simple test with a local MySQL database:
  - Reading is 5x slower than with ROOT I/O
  - Writing is functional but requires significant optimization of the code.
- Current implementation of the SQL communication (text oriented) could be greatly improved.
  - Could use some expertise in MySQL and odbc (to reinvigorate RDBC)

TTree Draw

Philippe Canal (FNAL)
2005 Root Workshop

TTree::Scan extensions

```
tree->Scan("a:b:c","","colsize=30 precision=3 col=::20.10");
```

- The output of TTree::Scan can now be customized via the 3rd argument
- Column size:
  - The default is 9 characters
  - It can be modified with "`colsize=ss`" where ss is the new size
- Floating point precision:
  - The default is 9 digits
  - It can be modified with "`precision=p`"
- Individual columns:
  - The size of each columns can be specified via "`col=xxx`"
  - Where 'xxx' is colon (:) delimited list of printing format for each column if no format is specified for a column, the default is
- Array elements
  - The number of array values printed per events can be restricted using "`lenmax=dd`" where 'dd' is the number of element printed

Looking at the Tree

```
myTree->Scan("fEvtHdr.fDate:fNtrack:fPx:fPy","",
   "colsize=13 precision=3 col=13:7::15.10");

***********************************************************************
* Row * Instance * fEvtHdr.fDate * fNtrack *           fPx *             fPy *
***********************************************************************
*   0 *        0 *        960312 *     594 *          2.07 *     1.459911346 *
*   0 *        1 *        960312 *     594 *         0.903 *   -0.4093382061 *
*   0 *        2 *        960312 *     594 *         0.696 *    0.3913401663 *
*   0 *        3 *        960312 *     594 *        -0.638 *     1.244356871 *
*   0 *        4 *        960312 *     594 *        -0.556 *   -0.7361358404 *
*   0 *        5 *        960312 *     594 *         -1.57 *   -0.3049036264 *
*   0 *        6 *        960312 *     594 *        0.0425 *    -1.006743073 *
*   0 *        7 *        960312 *     594 *          -0.6 *    -1.895804524 *
```

TTree::Draw extensions

- @ notation

```
tree->Draw("event.fTracks.size()");  // Size of the tracks
tree->Draw("event.@fTracks.size()"); // Number of tracks
```

- Sum$
  - Return the sum of the value of the elements of the formula given as a parameter.

```
tree->Draw("Sum$(formula)/Length$(formula)");
// Histo of the mean of ‘formula’ in each event
```

- Allow more cases of branch names
  - The TFormula parsing was improved to allow the branch names to contain class template names (aka myclass<int,double>)

TTree::Draw extensions

- TTree::Draw can call any function or member function which takes numerical arguments:

```
tree->Draw("TMath::Abs(event.fH.GetMean())");
```

- TTree::Draw can execute scripts in a context where the name of the branches can be used as a C++ variable.

```
// File hsimple.C
double hsimple()
{
   return px
};
```

```
// File track.C
double track()
{
   int ntrack = event->GetNTracks();
   if (ntrack>2) {
      return fTracks.fPy[2];
   }
   return 0;
};
```

```
tree->Draw("hsimple.C");
```

```
tree->Draw("track.C");
```

TTree::MakeProxy

- Enables ***`tree->Draw("hsimple.C");`***
- Generates a skeleton analysis class inheriting from TSelector and using TBranchProxy.
  - TBranchProxy is the base class of a hierarchy implementing an indirect access to the content of the branches of a TTree.
- Main Features:
  - **on-demand** loading of branches
  - ability to use the 'branchname' as if it was a data member
  - protection against array out-of-bound
  - ability to use the branch data as an **object** (when the user code is available)
  - Gives access to all the functionality of TSelector
- Example in $ROOTSYS/tutorials:

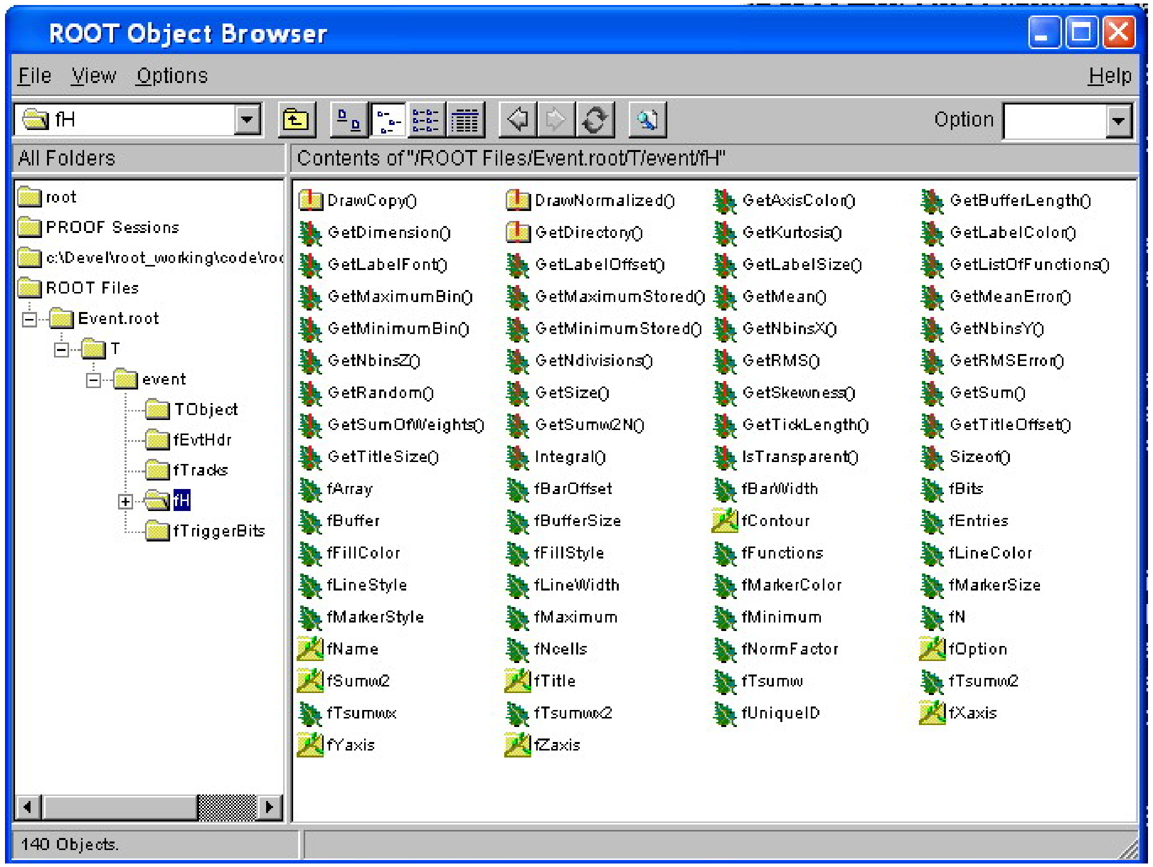
  h1analysisProxy.cxx , h1analysProxy.h and h1analysisProxyCut.C

TFormula Optimizations

- New implementation of the executor part of TFormula
  - Combines or replaces multiple operations by a single indirect function call.
  - Pre-calculate constant expressions
  - minimizes the size of the existing switch
- This result in a significant speed-up of the execution
  - Especially noticeable if used for minimization

Browsing extension

- Can now Browse:
  - Split objects
  - Unsplit objects
  - Collections
- And can now see
  - Simple member functions
  - Transient members
  - Persistent members



Contents of "/ROOT Files/Event.root/T/event/fH"

root
PROOF Sessions
c:\Devel\root_working\code\roo
ROOT Files
Event.root
T
event
TObject
fEvtHdr
fTracks
fH
fTriggerBits

DrawCopy()	DrawNormalized()	GetAxisColor()	GetBufferLength()
GetDimension()	GetDirectory()	GetKurtosis()	GetLabelColor()
GetLabelFont()	GetLabelOffset()	GetLabelSize()	GetListOfFunctions()
GetMaximumBin()	GetMaximumStored()	GetMean()	GetMeanError()
GetMinimumBin()	GetMinimumStored()	GetNbinsX()	GetNbinsY()
GetNbinsZ()	GetNdivisions()	GetRMS()	GetRMSError()
GetRandom()	GetSize()	GetSkewness()	GetSum()
GetSumOfWeights()	GetSumw2N()	GetTickLength()	GetTitleOffset()
GetTitleSize()	Integral()	IsTransparent()	Sizeof()
fArray	fBarOffset	fBarWidth	fBits
fBuffer	fBufferSize	fContour	fEntries
fFillColor	fFillStyle	fFunctions	fLineColor
fLineStyle	fLineWidth	fMarkerColor	fMarkerSize
fMarkerStyle	fMaximum	fMinimum	fN
fName	fNcells	fNormFactor	fOption
fSumw2	fTitle	fTsumw	fTsumw2
fTsumwx	fTsumwx2	fUniqueID	fXaxis
fYaxis	fZaxis		

140 Objects.

Upcoming Features

- Main focus: References
  - Will implement a TVirtualRefProxy providing a generic interface for reference objects (including *GetObject, GetObjectType*). This will be used by TTree::Draw to be able to dereference TRefs and pool::ref
- MakeProxy
  - Add support for STL containers
  - Add support for CINT-interpretation
- I/O
  - Variable size array of ‘Foreign’ Object: `Obj *fArr; //[n]`
- TTree
  - Indexing using bitmap algorithm (TBitMapIndex) from LBL (See John Wu’s talk)