Update concerning HPS Conditions Database

H. Neal

21 July 2011

Writing Objects

```
// Prepare a transient object and a factory which would create a persistent object
// out of the transient one.
  const unsigned int objectSize = 32;
  std::vector<int> data( objectSize );
  for( int i = 0; i < objectSize; ++i ) data[i] = i;

  CdbRooTestClassVarSize transientObject( "My first object stored in teh database.", data );
  CdbRooTObjectFactory< CdbRooTestClassVarSize, CdbRooTestClassVarSizeR > objectFactory( transientObject );

// Store a new object in the database.

  CdbObjectPtr objectPtr;

  if( CdbStatus::Success != conditionPtr->storeObject( objectFactory,
                                                       beginValidity,
                                                       endValidity,
                                                       objectPtr )) {
      cerr << "error: failed to store the object in the condition: " << conditionName << endl;
      return 1;
  }

// Display results.
  cout << "<OBJECT>" << endl
       << "  <BEGIN 'VISIBLE'  VALIDITY> " << CdbTimeUtils::time2string2( objectPtr->begin( )) << "\n"
       << "  <END   'VISIBLE'  VALIDITY> " << CdbTimeUtils::time2string2( objectPtr->end( )) << "\n"
       << "  <BEGIN 'ORIGINAL' VALIDITY> " << CdbTimeUtils::time2string2( objectPtr->beginOriginal( )) << "\n"
       << "  <END   'ORIGINAL' VALIDITY> " << CdbTimeUtils::time2string2( objectPtr->endOriginal( )) << "\n"
       << "  <CREATED>                   " << CdbTimeUtils::time2string2( objectPtr->inserted( )) << "\n"
       << "  <ID>                        " << objectPtr->id( ) << "\n"
       << "  <LEGACY ID>                 " << objectPtr->legacyId( ) << "\n"
       << "  <TYPE>                      " << objectPtr->type( ) << "\n";
```

SVT Example

```
// store wafer alignments
  if(_svtwaferaligns.value() != std::string("None")){
    CdbLoadList
      loadwafer("svt",_wafAlignName.value().c_str(),::createWafer);
    CdbStatus wstat =
loadwafer.storeObjects(_svtwaferaligns.pathname().c_str());
    if(wstat != CdbStatus::Success){
      cerr << name() << " Error loading wafer alignments: aborting" << endl;
      assert(false);
    } else if (verbose())
      cout << name() << " Successfully loaded wafer alignments " << endl;
  }
```

Loading an Object

```
  CdbTransaction transaction( CdbTransaction::Read );

// Find the specified condition in the database
  CdbConditionPtr conditionPtr;
  if( CdbStatus::Success != ( result = CdbCondition::instanceFromAny( conditionPtr,
                                                                      conditionName ))) {
      cerr << "error: failed to find the condition '" << conditionName << "' because of: " << result << ".\n";
      return 1;
  }

// Find an object in the condition at teh specified point of teh validity timeline
// assuming that a proper configuration of the condition exists in the default view
// of the database.

  CdbObjectPtr objectPtr;

  if( CdbStatus::Success != conditionPtr->findObject( objectPtr,
                                                      atValidity )) {
      cerr << "error: failed to find a object in the condition: " << conditionName << endl;
      return 1;
  }

// Display metadata information
  cout << "<OBJECT>" << endl
       << "  <BEGIN 'VISIBLE'  VALIDITY> " << CdbTimeUtils::time2string2( objectPtr->begin( )) << "\n"
       << "  <END   'VISIBLE'  VALIDITY> " << CdbTimeUtils::time2string2( objectPtr->end( )) << "\n"
       << "  <BEGIN 'ORIGINAL' VALIDITY> " << CdbTimeUtils::time2string2( objectPtr->beginOriginal( )) << "\n"
       << "  <END   'ORIGINAL' VALIDITY> " << CdbTimeUtils::time2string2( objectPtr->endOriginal( )) << "\n"
       << "  <CREATED>                   " << CdbTimeUtils::time2string2( objectPtr->inserted( )) << "\n"
       << "  <ID>                        " << objectPtr->id( ) << "\n"
       << "  <LEGACY ID>                 " << objectPtr->legacyId( ) << "\n"
       << "  <TYPE>                      " << objectPtr->type( ) << "\n"
       << "  <TYPE KEY>                  " << objectPtr->typeKey( ) << endl;
```

Loading SVT conditions

```
 std::vector<item> m;
 m.push_back("/svt/SvtPGlobalAlign") ;
```

…

```
   int k=0;
   for (std::vector<item>::iterator i=m.begin();i!=m.end() && k<10;++i,++k) {
     const HepVector& p = i->par(t);
```

…

```
 cerr << "searching " << _name << " for " << time << endl;
 CdbObjectPtr oPtr;
 if( CdbStatus::Success != _p->findObject( oPtr, time )) {
         cerr << "Failed to find " << _name << " for time " << time <<endl;
         ::exit(1);
 }
```

Plan

- The current CDB code has almost no dependencies on the BaBar framework
  and Igor is eager to provide a product that is project independent
- Homer is working on putting conditions access into the HPS lcsim code
  - In this context the work should be able to proceed with no extra cost to HPS
    - Alternatives of temporary shift of percentages between Homer ↔ Igor are being investigated

CDB Transaction Preparation

```
  BdbTime beginValidity = BdbTime::minusInfinity;
  if( !string2time( beginValidity, argv[2] )) return 1;

  BdbTime endValidity = BdbTime::minusInfinity;
  if( !string2time( endValidity, argv[3] )) return 1;

// Make sure there is a valid transaction is started

  CdbTransaction transaction( CdbTransaction::Update );

// Find the specified condition in the database

  CdbConditionPtr conditionPtr;
  if( CdbStatus::Success != ( result = CdbCondition::instanceFromAny( conditionPtr,
                                                                      conditionName ))) {
      cerr << "error: failed to find the condition '" << conditionName << "' because of: " << result << ".\n";
      return 1;
  }

// Prepare a transient object and a factory which would create a persistent object
// out of the transient one.

  const unsigned int objectSize = 32;
  std::vector<int> data( objectSize );
  for( int i = 0; i < objectSize; ++i ) data[i] = i;

  CdbRooTestClassVarSize transientObject( "My first object stored in the database.", data );

  CdbRooTObjectFactory< CdbRooTestClassVarSize, CdbRooTestClassVarSizeR > objectFactory( transientObject );
```