Transient Conditions/Detector Store using StoreGate

ATLAS Software Workshop
EDM Session
Dec 6 2001

Charles Leggett <CGLeggett@lbl.gov>

Transient Detector Store

- new instance of StoreGate, which doesn't get wiped at end of each event:

```
ApplicationMgr.ExtSvc += { "StoreGateSvc/DetectorStore" };
```

```
StoreGateSvc *m_detectorStore;

StatusCode sc = service("DetectorStore", m_detectorStore);
```

- acronym: **TDeS** (TDS taken)
- filled once in some algorithm's initialize method

Transient Conditions Store Objectives

- acronym: **TCS**

- Store Conditions data in TCS using StoreGate
- Lifetime of objects in TCS are set by database - TCS should not be wiped at the end of each event
- User should not have to check validity of TCS object - handled behind the scenes
- Ability to use call backs is foreseen
- For objects that depend on multiple objects in TCS, ability to form associations so that when dependants get updated, composite object also gets updated
- Use converters to interface with ITDb Conditions Database

TDeS / TCS

- Why two separate stores for TCS and TDeS?
  - different caching policies
  - we are still prototyping, and is easier to collapse two separate stores into one than split one into two

- use patterns:
  - put detector geometry information in TDeS
  - put alignment information into TCS
  - link detector elements in TDeS with associated alignment data in TCS using an “AutoHandle”. “AutoHandle” can be associated with multiple objects in TCS, so that when one changes the “AutoHandle” knows to recalculate itself.

Implementation

- Using StoreGate and DataProxies

- Define a new type of proxy, named PullDataProxy (to distinguish from PushDataProxy)
  - has associated ValidityRange
  - validity checked each time proxy is dereferenced via a DataHandle
  - if invalid, uses standard persistency mechanisms to invoke converters, caches new validity range and retrieved object
  - requires DBValidityRange class, and IDBOpaqueAddress class with access mechanisms to check/set validity range
- Users specify a complete DataHandle:
  - `DataHandle<DATA, DataHandleIterator<PullDataProxy> >`
  - `typedef to TimedDataHandle`
- Retrieved with a key
  - `StoreGateSvc->retrieve(TimedDataHandle,key=identifier)`

PullDataProxy.h

```
class PullDataProxy : public DataProxy
{
 protected:
  typedef DBValidityRange::tstamp_type tstamp_type;

 public:
  // Constructors
  PullDataProxy();                     // default constructor
  PullDataProxy(DataObject*, const DBValidityRange&, const std::string& key);
  PullDataProxy(IOpaqueAddress* pAddress, const std::string& key);

  // Destructor
  virtual ~PullDataProxy();

  //IRegistry implementation inherited from DataProxy
  IOpaqueAddress* address() const;

 //own stuff
  bool isValid() const;
  static void setCurrent(const tstamp_type& current) { m_current = current; }
  void setAddress(IOpaqueAddress*); // dcast and set DBValidityRange

 protected:
  const tstamp_type& getCurrent() const { return m_current; }

 private:
  IDBOpaqueAddress* p_DBaddress;
  DBValidityRange m_range;
  static tstamp_type m_current;
};
```

DBValidityRange.h

```
class DBValidityRange {
 public:
  typedef EventID tstamp_type;
   DBValidityRange() :
    m_begin(0), m_end(0) {}
  DBValidityRange(const tstamp_type* begin, const tstamp_type* end) {
    setRange(begin, end);
  }
  inline bool isInRange(const tstamp_type& current) const {
    return (0 != m_begin && 0 != m_end);
  }
  void setRange(const tstamp_type* begin, const tstamp_type* end) {
    if (0 == begin || 0 == end || begin>end) {
      m_begin=0; m_end=0;
    } else {
      m_begin=begin; m_end=end;
    }
  }
 private:
  const tstamp_type* m_begin;
  const tstamp_type* m_end;
};
```

DBOpaqueAddress

```
class IDBOpaqueAddress : public IOpaqueAddress {
 public:
  virtual void setValidityRange(const DBValidityRange&)=0;
  virtual const DBValidityRange& getValidityRange() const = 0;
};
```

```
class DBOpaqueAddress : public virtual IDBOpaqueAddress,
                        public virtual GenericAddress
{
public:
  DBOpaqueAddress() {};

  virtual void setValidityRange(const DBValidityRange& range);
  virtual const DBValidityRange& getValidityRange();
private:
  DBValidityRange m_range;
};
```