An RTAG View of Event Collections, and Early Implementations

David Malon

malon@anl.gov

ATLAS Database Group

LHC Persistence Workshop

5 June 2002

Event Collection Types

- Implicit (by containment):
  - “whatever events are in this file, or in this sequence of files”
- Explicit:
  - “this specific list of events”
  - may consist of selected events from this file, a few from that file, …
- Project should support both kinds of collections
- RTAG also proposed that explicit collections be queryable (like tag databases)

Implicit collections

- Assumption: Implicit collections are created implicitly--simply writing a sequence of events results in an event collection
- Providing the file or (file list) as input should suffice to support iteration
- Collection may be named and cataloged, but this is an additional, optional, step
- Input event iterator interface should be the same as for explicit collections

Explicit collections

- Equivalent to a list of "pointers" to events
- Because the project will deliver externalizable Refs encapsulating possibly technology-specific persistent addresses, RTAG proposal is an implementation that uses these Refs
- Explicit collections behave like a variable-length list of Refs to experiment-specific event entry points
- Collections are created explicitly, either by experiment's framework or by user: Ref to event entry point is inserted into a collection
- Natural starting point for an input interface, then, is something like an STL input iterator
- Is this a reasonable common input interface for both explicit and implicit collections?

Queryable collections

- In an explicit collection, optional "tag" data may also be associated with an event
- Ref + tag data are inserted into an explicit collection
- Mental model is of a relational table whose columns are the tag attributes, with one column containing a Ref to the event
- Propose to implement explicit collections both in the ROOT layer and in the relational layer
- Resulting collection would allow preselection of events of interest, but could also be used directly for analysis
- An architectural view is that tags are data exported from the event (event-level metadata)

Tag specification interface

- Tag definition requires provision of a list of property names and types
- Many projects and technologies have proposed interfaces (generic tags, addItem methods for ntuples, SQL CREATE TABLE syntax, XML, …)
- Specific choice is perhaps not so important, but interface should be the same for event tags, collection metadata, file metadata, …
- Propose to support types in the approximate intersection of MySQL types and ROOT types

Query interface

- Should not invent a new query language
- Propose a strict, very limited subset of SQL-xx, likely to be supportable in most technologies
  - Predicates applied to single tags
  - Predicates are boolean combinations of range queries on attributes
  - Comparison operators (<, >, =, !=, …) applied to single attribute (column) names—no arithmetic
  - Logical operators (AND, OR, NOT), and grouping ()

Collection services for the common project

- Early project implementation of collections allows us to support a user view of input/output specification at a level "higher" than files, even when our implementation is file-based
- Relational implementations allow the project to do early prototyping of facilities that make nontrivial use of relational capabilities: querying, indexing, server-side selection, …

A sample scenario

- A production job produces an explicit (tag) collection, instantiated in a ROOT file
- N production jobs produce N such files
- A concatenation step produces an explicit union of these tag tables to create a SINGLE relational table, which is indexed to support fast SQL predicate-based selection

Collection-level operations

- Collection creation, naming, registration
- Subset selection (satisfying SQL predicate)
- Copying
  - E.g., from one technology to another
- Unions—declarative, and explicit concatenation
- …other collection operations and services as required in later releases

Optional potential extensions

- Most physics processing is "for each:" "for each event that satisfies my condition, do …"
  - Order is unimportant, as long as all qualifying events are processed
- A U.S. Grand Challenge Project in support of the STAR experiment delivered order-optimized iteration
- Simplified view:
  - sort events that satisfy your condition into groups according to the file(s) they are in
  - deliver first the groups of events whose files are already cached
  - initiate prefetching of files for events whose data are not already in the cache (remote? On tape?)
- Step one should be "easy" if we can map Refs to file ids; the rest can come later (…or not…)

Volunteering…

- Argonne/ATLAS is prepared to volunteer...
  - to deliver initial implementations of event collections and collection services in a reasonable timeframe (this summer?)
  - to ensure that deliverables meet reasonable requirements of the four experiments
  - to do prototyping/benchmarking of relational capabilities (indexing, …)
  - to partner with others with similar interests
- It is clear that many others have done related work (LHCb, BaBar, ROOT team, IT/DB, DESY, …); we are interested partly because the work is valuable to us--ATLAS has not done tag database prototyping to date