SQLBuilder: a replacement metadata language to SQL translator for SAM

Randolph J. Herber*, FNAL, Batavia, IL 60510, USA

Abstract

SQLBuilder is a replacement metadata language to SQL translator for SAM. The exsting translator has been in use for five years and needs replacement because of systemic problems, including desired enhacements.

OVERVIEW

SAM [1] is in use at CDF, DØ and MINOS. It uses a metadata language to write *dataset definitions* (rules) to select files for *snapshots* (lists of files which meet a dataset definition at the time the snapshot was created, are stored in the data base and normally are not changed afterward). The reasons for this metadata language are: the users want to avoid the complexities of SQL and they have demonstrated that they can easily overload a data base system with improperly written SQL. The present metadata language to SQL translator suffers from an ad-hoc tokenizer, an ad-hoc parser and internal data structures that do not lend themselves to easy processing, easy maintenance or effective storage of control data. SQLBuilder is intended to be a replacement metadata language to SQL translator for SAM.

SQLBuilder's purpose is to address the aforementioned problems in the present metadata language translator. The target language will be Oracle 9i or 10g SQL [2]. The generated queries will return sets of file identifiers. CORBA [3] will be used to communicate with the remainder of the SAM system.

The selection criteria are named, parameterized tests on the SAM metadata combined by boolean operators. Intermediate result sets may be combined by set operators. There will be a capacity to include previous SAM dataset definitions (with a check for infinite inclusion recursion) via the metadata language operator `__SET__`.

LESSONS LEARNED

Bad ideas:

- Using hyphen both for negative signs for numbers and for a range separator. This requires parser-generators with two token look-ahead or very specialized handling in the tokenizer.
- Implicit 'and' or '='. Optional elements in a grammar are hard to handle and can cause ambiguities.
- Unquoted string values complicate recognization of keywords.
- Parameters with only string values that only can be compared for equality.
  Parameters should be data typed for strings, numerics and times and be comparable for all the relational operators.
- Parameters should be usable with data definitions and other entities.
  In SAM, parameters are associated only with files and processing requests.
- File names are not versioned, time stamped or otherwise marked. This is needed so that bad files can be replaced without destroying the historical value of the data base by removing metadata from the data base. It would allow snapshots to show where the old file versions were used. This versioning may cause changes in the metadata language. File name alone is not sufficient to identify a file.
- Storing the data definition statement in an intermediate form.
  The original data definition string needs to be stored in the data base to document what the user originally specified and to support `__SET__`.
- The use of multiple `OR` operators very rapidly results in poor data base performance.
- The use of the `MINUS` operator, in our experience, on Oracle data bases at least, has very poor performance.

Good ideas:

- Using a parser-generator is a good idea as it is extremely difficult to write or maintain one correctly by hand. LL [4] parsers are somewhat easier than LR [5] parsers to write by hand; but, they are somewhat less efficient and good error message generation is somewhat more difficult.
- Using UNIX style wild cards instead of Oracle style wild cards in string values. Users do expect to use '?' and '*' and do not expect to use '_' and '%' as wildcards.
- Using multiple valued tests instead of multiple single valued tests is a good idea. Consider, e.g., a typed run number which occurs both in MINOS and at DØ: `run(200000 'monte_carlo')` vs. `run_number = 200000 and runtype = 'monte_carlo'`. The first form clearly indicates that both conditions on one chain of data base tables. This eases implementation significantly [6].
- Parameters to associate arbitary attributes to entities such as files and requests.

* herber@fnal.gov, +1.630.840.2966, Mail Stop 318

- The program's services also should be available via Web services as well as CORBA.

DIMENSIONS

There are several classes of *dimensions* (tests on the data base that can be used construct data definitions):

- Tests on single columns within the database. E.g., `file_name = 'fred'` or `snapshot_id = 83467`.
- Tests on multiple columns in one or more tables along a joined chain of tables within the database. E.g., `run(200000 'monte_carlo')`.
- Tests on parameters. E.g., `pythia.topmass='137'`.
- Tests specified by SQL functions or code on multiple columns in one or more tables along a joined chain of tables within the database.
- Test data sources should be specified by data base table and column pairs and *Chains and Links*, which are *links* (description of the join conditions between two tables) and *chains* (list of links needed to connect from the search target data base table (in the SAM implementation, that is the table `DATA_FILES`) to the furthest table needed for the test.

IMPLEMENTATION PLAN

SQLBuilder is to be written in Java 5 [7]. Parser-generators, such as Antlr or JavaCC and JJTree will used to write the "front-end". After the *AST* [8] (**A**bstract **S**yntax **T**ree) has been built with appropriate attributes given to the nodes of the tree, the tree will be processed, possibly several times, to rewrite the expression as needed and to gather information on what dimensions are used.

It will be necessary to analyze the present usage of the data definition language (35K at CDF, 326K at DØ and 1151 at MINOS) in terms of the language elements and constructs used. Test cases to test all language elements at least once will need to be added to the test suite.

The location of the Chains and Links (defined above) data storage is not yet decided. There are both technical and management level issues here. Proposed locations are:

- program code resident. This is undesirable for maintenance reasons.
- data base resident. This could be done with JDBC [9] or Hibernate [10] and would permit reasonably easy maintenance.
- Flat text file using a defined syntax. A separate tokenizer and parser pair would be written to read the file.
- XML [11] file or web page. In this case, the tokenizer and parser would be available in Java. A DOM [12] or XML editor would be needed to ease maintenance.

The back-end will consist of an AST to an abstract SQL translator followed by an abstract SQL translator to Oracle specific SQL translator. The AST to abstract SQL translator basically generates the joins (tables and where conditions) and values tests (where conditions) in a data base neutral form. The abstract SQL translator to Oracle specific SQL translator generates the SQL to be returned to the SAM DB Server.

Extensive testing will be necessary based on the use cases mentioned above.

User, system and administration level documentation are planned to be written.

The references contain the details of the proposed grammar [15], written in a EBNF [13] style and being essentially context-free [14] and the tokens [16].

ACKNOWLEDGEMENTS

- This project is sponsored by DOE contract DE-AC02-76CH03000.
- The Fermilab and the participating institutions staffs for their part in creating SAM.
- Adam L. Lyon, Krzysztof Genser, Doug Benjamin, Lauri L. Loebel-Carpenter, Sinisa Veseli and Stephen P. White assisted with this paper.

REFERENCES

[1] SAM: http://d0db-dev.fnal.gov/sam/

[2] Oracle^{tm} 9i: https://misdev1.fnal.gov/oracledoc/9iv2/DOC/index.htm

[3] CORBA^{tm}: http://www.omg.org/gettingstarted/corbafaq.htm.

[4] LL parser: http://en.wikipedia.org/wiki/LL_parser

[5] LR parser: http://en.wikipedia.org/wiki/LR_parser

[6] Dependence — Independence https://plone3.fnal.gov/SAMGrid/Members/herber/Independence-Dependence/file_view

[7] Java^{tm} computer language: http://java.sun.com/

[8] Abstract Syntax Tree http://en.wikipedia.org/wiki/Abstract_syntax_tree.

[9] JDBC^{tm}: http://java.sun.com/products/jdbc/.

[10] Hibernate — object persistence for Java http://www.hibernate.org/.

[11] XML: http://www.w3.org/XML/

[12] DOM: http://www.w3.org/DOM/

[13] Extended Backus-Norm Form: http://www.cl.cam.ac.uk/ mgk25/iso-14977.pdf

[14] Context-Free Grammars: http://www.cs.utsa.edu/ qingyi/cs4713/handouts/CFG.pdf

[15] Proposed metadata language grammar: https://plone3.fnal.gov/SAMGrid/Members/herber /Grammar/file_view.

[16] Proposed metadata language tokens: https://plone3.fnal.gov/SAMGrid/Members/herber/Tokens /file_view.

[17] Design guidelines: http://www.physics.gla.ac.uk/pipermail/metadata/2004-September/000069.html.