TRPL - Feature #1755

implement database-backed filtering/walking optimization

10/29/2012 06:31 PM - Greg Shah

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	160.00 hours
Target version:	Database Backed Storage		
billable:	No	**vendor_id:**	GCD

Description

Related issues:

Blocked by TRPL - Feature #1759: design a database to back all TRPL processing	**New**

History

#1 - 10/29/2012 07:08 PM - Greg Shah

- File IMG_20120911_174539.jpg added

- File IMG_20120911_174547.jpg added

- File IMG_20120911_174551.jpg added

Some changes are to the pattern engine/runtime and some are in the TRPL compiler (to emit the filtering expressions differently and so forth); the primary problem seems to be an ordering issue (how to duplicate the exact order of the node walking when one is not actually traversing all nodes).

Thoughts:

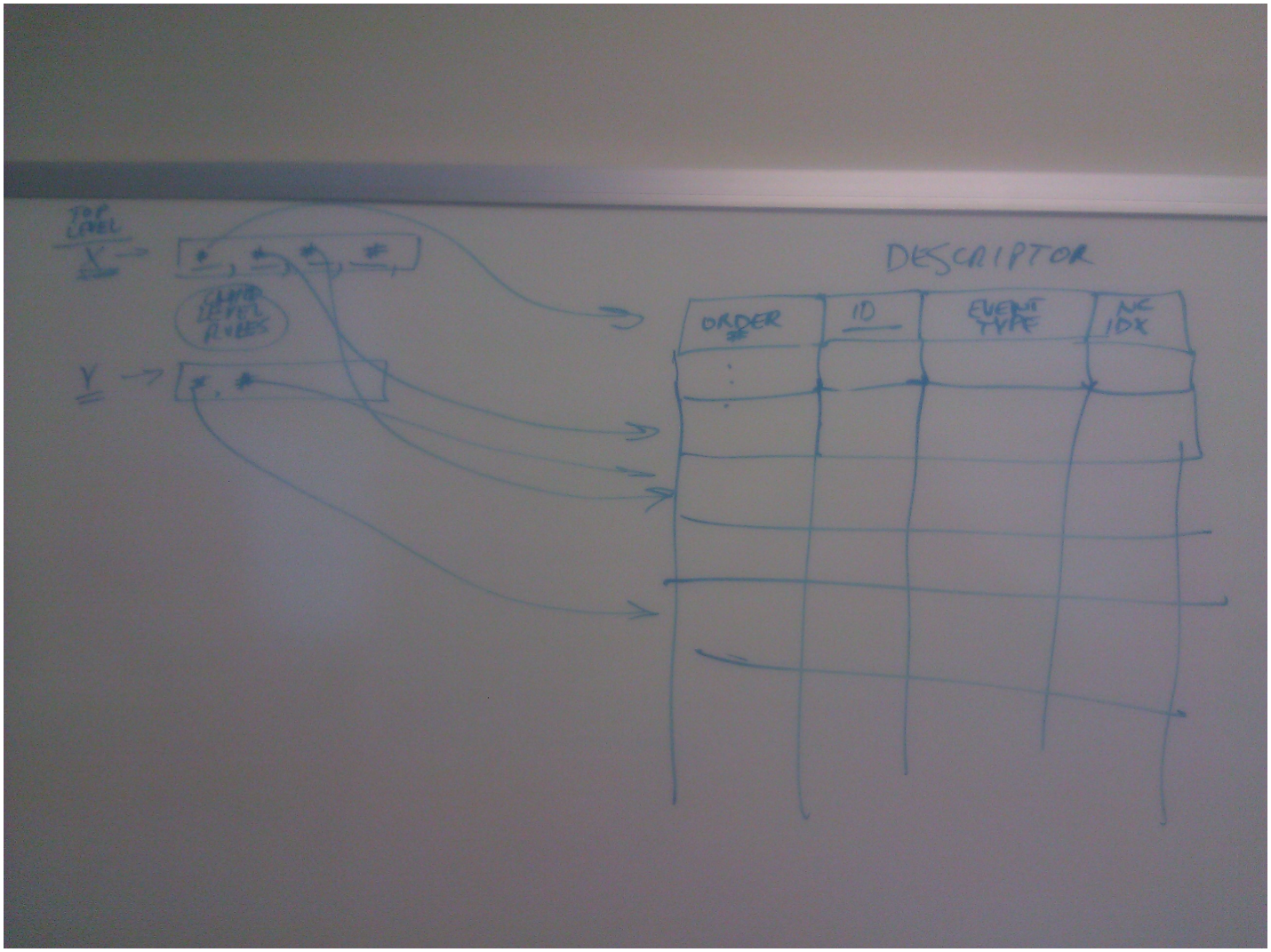
0. All nodes in the tree can be flattened into a degenerate list whose order is the same as the tree walk. Each entry in this list could be a "descriptor" that includes its ordinal (the order # which specifies its location in the list), the node ID, the event type (init, walk, descent, next-child, ascent, post), the parent ID and its next-child index (its relative horizontal position within its parent). In other words, this encodes the entire structure of the tree in a table form. Implementation options include using this at the database level itself to encode the tree OR to create this in memory when the AST is "loaded" and then maintaining the in-memory version in lockstep with the database changes when any changes occur.

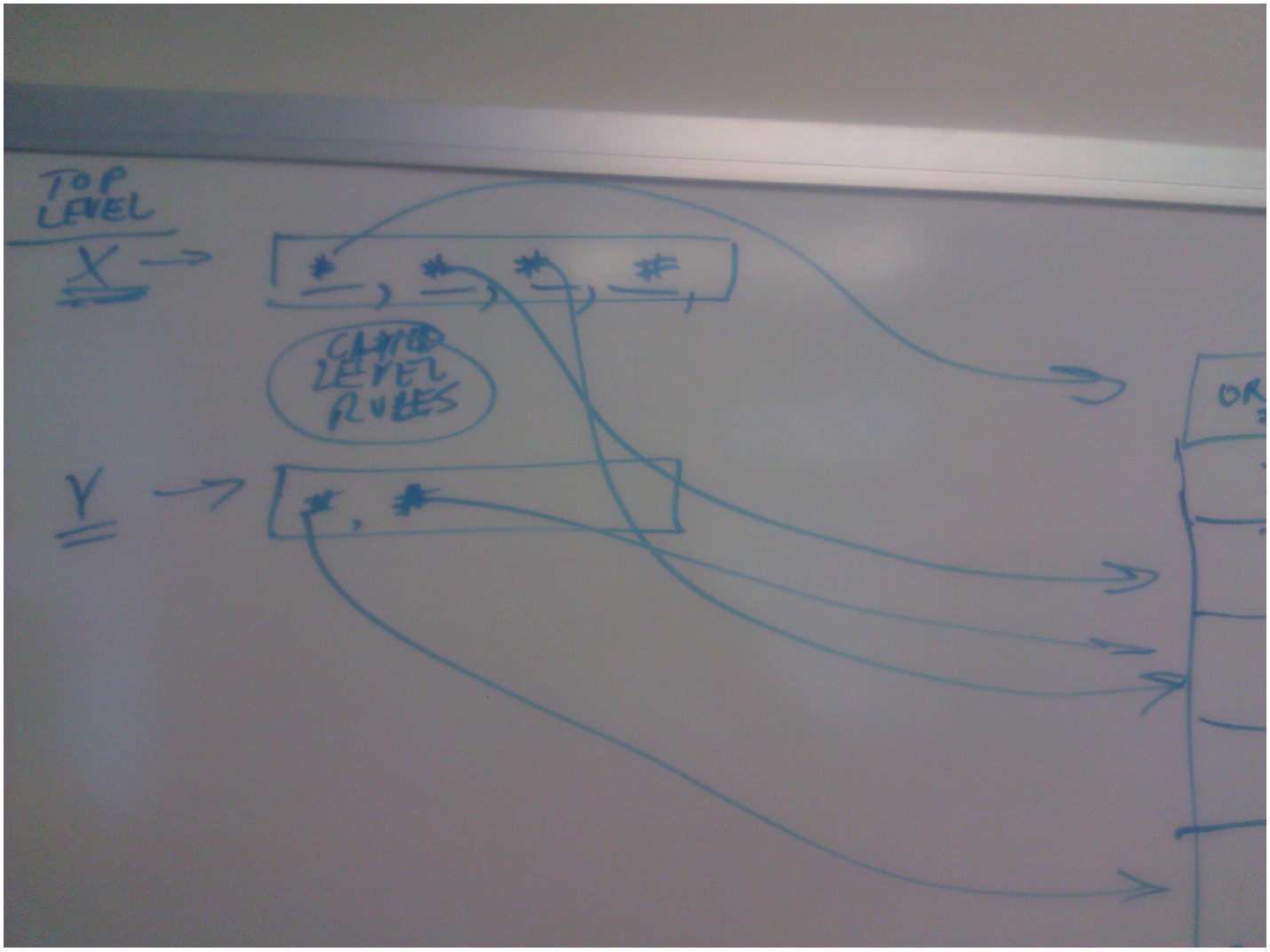
1. Each enclosing level of rules can be viewed as a filter, potentially reducing the walk (for any logic nested inside the rule) to a subset of the nodes of the enclosing level. The expressions in each level can be partially (sometimes fully) converted to SQL to make a list of potentially matching nodes. The query would also take into account the event type of the rule being processed. This list of potentially matching nodes would essentially be a list of the ordinal numbers that reference descriptors (see #0 above). The idea is that you want to eliminate as many expression executions as is possible without changing the program flow. Filter rules MUST not have side-effects. But otherwise, this is reasonable to achieve. And initial results suggest that the performance improvement of this idea may be massive.

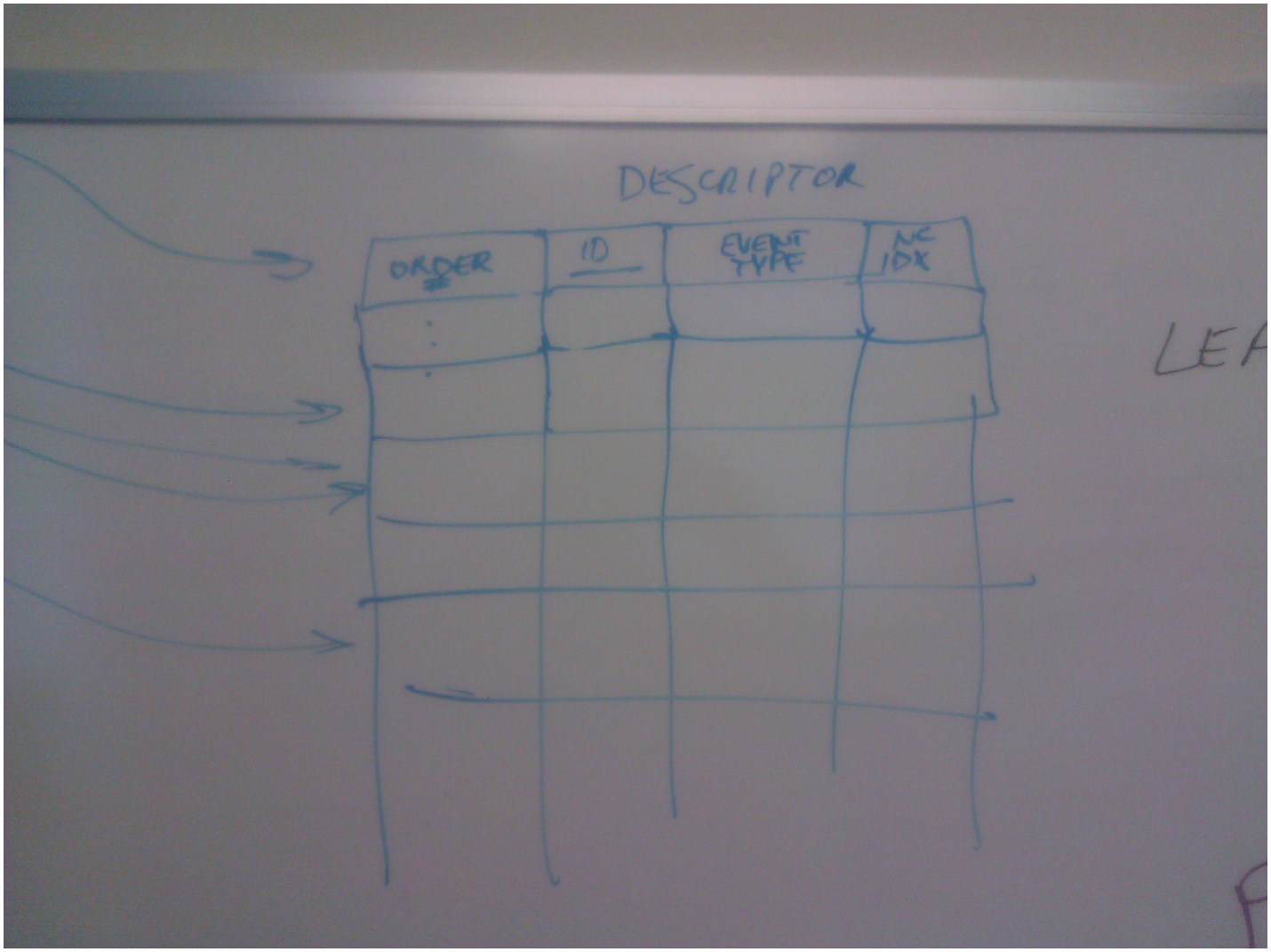
3. Every rule at the same level (e.g. all top-level rules, all 2nd level rules) would have to be executed in the correct order based on the potential descriptor matches. A rule that is closer to the top of the file will execute first if the same node appears in the potential match lists. A lower ordinal node (earlier in the tree walk) that appears in a potential match list for a rule that is lower in the file (but at the same level as other rules above which don't reference that node) will be found by that lower rule before higher rules will find higher ordinal nodes. In other words, the order of processing of the walk must be preserved within each level of rules and the ordinals can be used to do this in a very simple manner.

4. Once the rule is known to need to be executed, the remaining parts (which couldn't be turned to SQL) must be executed in TRPL (instead of SQL in the database) for the descriptor that is known to need processing next. If a match occurs, then the immediately enclosed rule (the next nested level) will be processed for matches with that node. It is possible to recursively use the approach in #1 and #2 above to further filter at nested rule levels. It is not clear when this may be a hindrance rather than a performance boon.

5. Another major performance benefit is to eliminate the constant and repeated XML parsing of the ASTs. Caching must be done in a smart manner to ensure that the database doesn't become a similar I/O bottleneck.


TOP LEVEL
DESCRIPTOR
ORDER #
ID
EVENT TYPE
NC IDX


TOP LEVEL
X
CHILD LEVEL RULES
Y


DESCRIPTOR
ORDER #
ID
EVENT TYPE
NC IDX

#2 - 10/31/2012 01:00 PM - Greg Shah

- Target version set to Code Improvements

#3 - 11/16/2016 09:59 AM - Greg Shah

- Target version changed from Code Improvements to Database Backed Storage

Files

IMG_20120911_174539.jpg	721 KB	10/29/2012	Greg Shah
IMG_20120911_174547.jpg	702 KB	10/29/2012	Greg Shah
IMG_20120911_174551.jpg	681 KB	10/29/2012	Greg Shah