Item 74
git_comments:

1. * Checking for boosted here because the request context will not have the elevated docs * until after the query is constructed. So to be sure there are no elevated docs in the query * while checking the cache we must check the request context during the call to hashCode().

git_commits:

1. **summary:** SOLR-5624: check for elevated documents in hashCode()
   message: SOLR-5624: check for elevated documents in hashCode() git-svn-id: https://svn.apache.org/repos/asf/lucene/dev/branches/branch_4x@1567649 13f79535-47bb-0310-9956-ffa450edef68

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Enable QueryResultCache for CollapsingQParserPlugin
   description:

jira_issues_comments:

1. Initial patch created from trunk, lightly tested. Looks good.
2. New patch. This patch provides logic that will cause caching to fail when elevated docs are present.
3. Added new patch just expanding one of the tests.
4. Commit 1566071 from [~jbernste] in branch 'dev/trunk' [https://svn.apache.org/r1566071] SOLR-5624: Enable QueryResultCache for CollapsingQParserPlugin
5. Commit 1566122 from [~jbernste] in branch 'dev/branches/branch_4x' [https://svn.apache.org/r1566122] SOLR-5624: Enable QueryResultCache for CollapsingQParserPlugin
6. Commit 1566309 from [~jbernste] in branch 'dev/trunk' [https://svn.apache.org/r1566309] SOLR-5624: Guard against NPE during cache warming
7. Commit 1566312 from [~jbernste] in branch 'dev/branches/branch_4x' [https://svn.apache.org/r1566312] SOLR-5624: Guard against NPE during cache warming
8. Commit 1567640 from [~jbernste] in branch 'dev/trunk' [https://svn.apache.org/r1567640] SOLR-5624: check for elevated documents in hashCode()
9. The previous commit checks for elevated documents in the query's hashCode() method. This needs to be done because elevated documents do not appear in the request context until after the query is constructed. This is because the QueryElevationComponent adds the elevated documents to the request context after the original query is constructed. When checking for the query in the QueryResultCache, we need to take into account the presence of elevated documents. The check that was added to the hashCode() method does this. We still need to look for elevated docs in the getFilterCollector() method in case caching was turned off.
10. Commit 1567649 from [~jbernste] in branch 'dev/branches/branch_4x' [https://svn.apache.org/r1567649] SOLR-5624: check for elevated documents in hashCode()
11. **body:** Hi Joel, I sent you an email but I'm not sure if you received it or not. I ran into a bit of trouble using the CollapsingQParserPlugin with elevated documents. To explain it simply, I want to exclude grouped documents when one of the members of the group are contained in the elevated document set. I'm not sure this is possible currently because as you explain above elevated documents are added to the request context after the original query is constructed. To try to better illustrate the problem. If I have 2 documents docid=1 and docid=2 and both have a groupid of 'a'. If a grouped query scores docid 2 first in the results but I have elevated docid 1 then both documents are shown in the results when I really only

want the elevated document to be shown in the results. Is this something that would be difficult to implement? Any help is appreciated.
label: code-design

12. I think the solution would be to remove the documents from liveDocs that share the same groupid in the getBoostDocs() function. Let me know if this makes any sense. I'll continue working towards a solution in the meantime. {code} private IntOpenHashSet getBoostDocs(SolrIndexSearcher indexSearcher, Set<String> boosted) throws IOException { IntOpenHashSet boostDocs = null; if(boosted != null) { SchemaField idField = indexSearcher.getSchema().getUniqueKeyField(); String fieldName = idField.getName(); HashSet<BytesRef> localBoosts = new HashSet(boosted.size()*2); Iterator<String> boostedIt = boosted.iterator(); while(boostedIt.hasNext()) { localBoosts.add(new BytesRef(boostedIt.next())); } boostDocs = new IntOpenHashSet(boosted.size()*2); List<AtomicReaderContext>leaves = indexSearcher.getTopReaderContext().leaves(); TermsEnum termsEnum = null; DocsEnum docsEnum = null; for(AtomicReaderContext leaf : leaves) { AtomicReader reader = leaf.reader(); int docBase = leaf.docBase; Bits liveDocs = reader.getLiveDocs(); Terms terms = reader.terms(fieldName); termsEnum = terms.iterator(termsEnum); Iterator<BytesRef> it = localBoosts.iterator(); while(it.hasNext()) { BytesRef ref = it.next(); if(termsEnum.seekExact(ref)) { docsEnum = termsEnum.docs(liveDocs, docsEnum); int doc = docsEnum.nextDoc(); if(doc != -1) { //Found the document. boostDocs.add(doc+docBase); *// HERE REMOVE ANY DOCUMENTS THAT SHARE THE GROUPID NOT ONLY THE DOCID //* it.remove(); } } } } return boostDocs; } {code}
13. [~dboychuck]: please post your question to the solr-user list, or (if you have an improvement you'd like to contribe) open a new Jira issue to track it. posting a comment in a resolve issue like this is almost certain to get lost and overlooked.