ALF-13260 – MySQL performance issue with select_ChildAssocOfParentByName

Status: Closed
Project: Alfresco
Component: Repository
Affects Version: 4.0.d Community
Fix Version: 4.1 Enterprise
Security Level: External
Environment: Debian Squeeze, Tomcat 6, MySQL 5.1.49

1. Problem description

In Alfresco Community 4.0.d running on Debian Squeeze with MySQL 5.1.49, heavy operations such as importing or creating thousands of documents can cause a serious performance degradation.

After such a load, the query used to resolve child associations by name, select_ChildAssocOfParentByName, starts taking several seconds (5–6 seconds) instead of milliseconds. Since this query is executed once for each item in a folder listing, overall page response time becomes unacceptable.

Inspection of the MySQL execution plan shows that the optimizer sometimes chooses a poor plan and does not use the most appropriate index on alf_child_assoc, especially when statistics are out of date or skewed by recent bulk inserts.

2. Typical symptoms

• High CPU usage on the MySQL server during navigation or content updates.
• Slow folder browsing and very slow content creation or property updates.
• MySQL slow query log containing select_ChildAssocOfParentByName with execution times of several seconds.
• EXPLAIN for the query showing a full scan or suboptimal index usage on alf_child_assoc.

3. Workaround: force index usage on alf_child_assoc

One effective workaround is to override the SQL mapping so that MySQL is explicitly instructed to use an appropriate index on alf_child_assoc. Example override for node-common-SqlMap.xml for the MySQL InnoDB dialect:

<sql id="select_ChildAssoc_FromSimple_By_Parent_Index"> from alf_child_assoc assoc use index (idx_alf_cass_pri) join alf_node parentNode on (parentNode.id = assoc.parent_node_id) join alf_store parentStore on (parentStore.id = parentNode.store_id) join alf_node childNode on (childNode.id = assoc.child_node_id) join alf_store childStore on (childStore.id = childNode.store_id) </sql> <select id="select_ChildAssocOfParentByName" parameterType="ChildAssoc" resultMap="result_ChildAssoc"> <include refid="alfresco.node.select_ChildAssoc_Results"/> <include refid="alfresco.node.select_ChildAssoc_FromSimple_By_Parent_Index"/> where parentNode.id = #{parentNode.id} and assoc.child_node_name = #{childNodeName} and assoc.child_node_name_crc = #{childNodeNameCrc} </select>

This change forces MySQL to use the specified index for the association lookup. Several installations reported stable performance for weeks after applying this patch.

4. Alternative: create a dedicated index and hint MySQL

In some large repositories, creating an additional index on child_node_name and child_node_name_crc and then forcing MySQL to use that index provides better performance. Example index creation on alf_child_assoc:

CREATE INDEX idx_alf_cass_pri2 ON alf_child_assoc (child_node_name, child_node_name_crc);

The SQL mapping can then be adjusted to use the new index:

<sql id="select_ChildAssoc_FromSimple_By_Parent_Index"> from alf_child_assoc assoc use index (idx_alf_cass_pri2) join alf_node parentNode on (parentNode.id = assoc.parent_node_id) join alf_store parentStore on (parentStore.id = parentNode.store_id) join alf_node childNode on (childNode.id = assoc.child_node_id) join alf_store childStore on (childStore.id = childNode.store_id) </sql>

Reports from production environments with more than a million documents indicate that this approach reduces per-query execution time from around 5 seconds to well under a second, restoring acceptable performance for content creation and updates.

5. Core fix: include type_qname_id to leverage composite index

There is a composite index on alf_child_assoc that includes parent_node_id, type_qname_id, child_node_name_crc, and child_node_name. However, the original query did not restrict type_qname_id, which prevented MySQL from fully using this index. The effective long-term fix is to include type_qname_id in the WHERE clause. Conceptually, the change is: Original pattern:

<select id="select_ChildAssocOfParentByName" parameterType="ChildAssoc" resultMap="result_ChildAssoc"> <include refid="alfresco.node.select_ChildAssoc_Results"/> <include refid="alfresco.node.select_ChildAssoc_FromSimple"/> where parentNode.id = #{parentNode.id} and assoc.child_node_name = #{childNodeName} and assoc.child_node_name_crc = #{childNodeNameCrc} </select>

Updated pattern:

<select id="select_ChildAssocOfParentByName" parameterType="ChildAssoc" resultMap="result_ChildAssoc"> <include refid="alfresco.node.select_ChildAssoc_Results"/> <include refid="alfresco.node.select_ChildAssoc_FromSimple"/> where parentNode.id = #{parentNode.id} and assoc.type_qname_id = #{typeQNameId} and assoc.child_node_name = #{childNodeName} and assoc.child_node_name_crc = #{childNodeNameCrc} </select>

By constraining type_qname_id, the optimizer can make use of the existing composite index, which dramatically improves performance without relying on explicit index hints. Internal testing for large-scale site creation scenarios showed a reduction from tens of seconds per site creation to approximately 150 ms.

6. Validation and recommended approach

After applying the fix that includes type_qname_id in the query, performance tests on later builds (4.0.2(.3) and 4.1.0) no longer showed slow queries for select_ChildAssocOfParentByName and overall system performance under heavy load was acceptable.

Recommended approach:

1. If you need an immediate workaround on an affected version, consider: • Adding an explicit index hint as shown in section 3, or • Creating a dedicated index and forcing its usage as in section 4.

2. For a long-term solution, ensure that your Alfresco version includes the change that adds assoc.type_qname_id to the WHERE clause of select_ChildAssocOfParentByName, or apply an equivalent patch.

3. After any change, run EXPLAIN on the query and monitor the MySQL slow query log under representative load to confirm that the appropriate index is being used and that execution time is acceptable.