Item 273
git_comments:

git_commits:

1. **summary:** PHOENIX-4616 Move join query optimization out from QueryCompiler into QueryOptimizer (addendum)
   message: PHOENIX-4616 Move join query optimization out from QueryCompiler into QueryOptimizer (addendum)

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Move join query optimization out from QueryCompiler into QueryOptimizer
   description: Currently we do optimization for join queries inside QueryCompiler, which makes the APIs and code logic confusing, so we need to move join optimization logic into QueryOptimizer. Similarly, but probably with a different approach, we need to optimize UNION ALL queries and derived table sub-queries in QueryOptimizer.optimize(). Please also refer to this comment: https://issues.apache.org/jira/browse/PHOENIX-4585?focusedCommentId=16367616&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16367616
   label: code-design
2. **summary:** Move join query optimization out from QueryCompiler into QueryOptimizer
   description: Currently we do optimization for join queries inside QueryCompiler, which makes the APIs and code logic confusing, so we need to move join optimization logic into QueryOptimizer. Similarly, but probably with a different approach, we need to optimize UNION ALL queries and derived table sub-queries in QueryOptimizer.optimize(). Please also refer to this comment: https://issues.apache.org/jira/browse/PHOENIX-4585?focusedCommentId=16367616&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16367616
   label: code-design
3. **summary:** Move join query optimization out from QueryCompiler into QueryOptimizer
   description: Currently we do optimization for join queries inside QueryCompiler, which makes the APIs and code logic confusing, so we need to move join optimization logic into QueryOptimizer. Similarly, but probably with a different approach, we need to optimize UNION ALL queries and derived table sub-queries in QueryOptimizer.optimize(). Please also refer to this comment: https://issues.apache.org/jira/browse/PHOENIX-4585?focusedCommentId=16367616&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16367616
   label: code-design
4. **summary:** Move join query optimization out from QueryCompiler into QueryOptimizer
   description: Currently we do optimization for join queries inside QueryCompiler, which makes the APIs and code logic confusing, so we need to move join optimization logic into QueryOptimizer. Similarly, but probably with a different approach, we need to optimize UNION ALL queries and derived table sub-queries in QueryOptimizer.optimize(). Please also refer to this comment: https://issues.apache.org/jira/browse/PHOENIX-4585?focusedCommentId=16367616&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16367616
   label: code-design
5. **summary:** Move join query optimization out from QueryCompiler into QueryOptimizer

description: Currently we do optimization for join queries inside QueryCompiler, which makes the APIs and code logic confusing, so we need to move join optimization logic into QueryOptimizer. Similarly, but probably with a different approach, we need to optimize UNION ALL queries and derived table sub-queries in QueryOptimizer.optimize(). Please also refer to this comment: https://issues.apache.org/jira/browse/PHOENIX-4585?focusedCommentId=16367616&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16367616
label: code-design

6. **summary:** Move join query optimization out from QueryCompiler into QueryOptimizer
   description: Currently we do optimization for join queries inside QueryCompiler, which makes the APIs and code logic confusing, so we need to move join optimization logic into QueryOptimizer. Similarly, but probably with a different approach, we need to optimize UNION ALL queries and derived table sub-queries in QueryOptimizer.optimize(). Please also refer to this comment: https://issues.apache.org/jira/browse/PHOENIX-4585?focusedCommentId=16367616&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16367616
   label: code-design
7. **summary:** Move join query optimization out from QueryCompiler into QueryOptimizer
   description: Currently we do optimization for join queries inside QueryCompiler, which makes the APIs and code logic confusing, so we need to move join optimization logic into QueryOptimizer. Similarly, but probably with a different approach, we need to optimize UNION ALL queries and derived table sub-queries in QueryOptimizer.optimize(). Please also refer to this comment: https://issues.apache.org/jira/browse/PHOENIX-4585?focusedCommentId=16367616&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16367616
8. **summary:** Move join query optimization out from QueryCompiler into QueryOptimizer
   description: Currently we do optimization for join queries inside QueryCompiler, which makes the APIs and code logic confusing, so we need to move join optimization logic into QueryOptimizer. Similarly, but probably with a different approach, we need to optimize UNION ALL queries and derived table sub-queries in QueryOptimizer.optimize(). Please also refer to this comment: https://issues.apache.org/jira/browse/PHOENIX-4585?focusedCommentId=16367616&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16367616
9. **summary:** Move join query optimization out from QueryCompiler into QueryOptimizer
   description: Currently we do optimization for join queries inside QueryCompiler, which makes the APIs and code logic confusing, so we need to move join optimization logic into QueryOptimizer. Similarly, but probably with a different approach, we need to optimize UNION ALL queries and derived table sub-queries in QueryOptimizer.optimize(). Please also refer to this comment: https://issues.apache.org/jira/browse/PHOENIX-4585?focusedCommentId=16367616&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16367616
10. **summary:** Move join query optimization out from QueryCompiler into QueryOptimizer
    description: Currently we do optimization for join queries inside QueryCompiler, which makes the APIs and code logic confusing, so we need to move join optimization logic into QueryOptimizer. Similarly, but probably with a different approach, we need to optimize UNION ALL queries and derived table sub-queries in QueryOptimizer.optimize(). Please also refer to this comment: https://issues.apache.org/jira/browse/PHOENIX-4585?focusedCommentId=16367616&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16367616
11. **summary:** Move join query optimization out from QueryCompiler into QueryOptimizer
    description: Currently we do optimization for join queries inside QueryCompiler, which makes the APIs and code logic confusing, so we need to move join optimization logic into QueryOptimizer. Similarly, but probably with a different approach, we need to optimize UNION ALL queries and derived table sub-queries in QueryOptimizer.optimize(). Please also refer to this comment: https://issues.apache.org/jira/browse/PHOENIX-4585?focusedCommentId=16367616&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16367616
12. **summary:** Move join query optimization out from QueryCompiler into QueryOptimizer

description: Currently we do optimization for join queries inside QueryCompiler, which makes the APIs and code logic confusing, so we need to move join optimization logic into QueryOptimizer. Similarly, but probably with a different approach, we need to optimize UNION ALL queries and derived table sub-queries in QueryOptimizer.optimize(). Please also refer to this comment: https://issues.apache.org/jira/browse/PHOENIX-4585?focusedCommentId=16367616&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16367616

13. **summary:** Move join query optimization out from QueryCompiler into QueryOptimizer
    description: Currently we do optimization for join queries inside QueryCompiler, which makes the APIs and code logic confusing, so we need to move join optimization logic into QueryOptimizer. Similarly, but probably with a different approach, we need to optimize UNION ALL queries and derived table sub-queries in QueryOptimizer.optimize(). Please also refer to this comment: https://issues.apache.org/jira/browse/PHOENIX-4585?focusedCommentId=16367616&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16367616
14. **summary:** Move join query optimization out from QueryCompiler into QueryOptimizer
    description: Currently we do optimization for join queries inside QueryCompiler, which makes the APIs and code logic confusing, so we need to move join optimization logic into QueryOptimizer. Similarly, but probably with a different approach, we need to optimize UNION ALL queries and derived table sub-queries in QueryOptimizer.optimize(). Please also refer to this comment: https://issues.apache.org/jira/browse/PHOENIX-4585?focusedCommentId=16367616&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16367616
15. **summary:** Move join query optimization out from QueryCompiler into QueryOptimizer
    description: Currently we do optimization for join queries inside QueryCompiler, which makes the APIs and code logic confusing, so we need to move join optimization logic into QueryOptimizer. Similarly, but probably with a different approach, we need to optimize UNION ALL queries and derived table sub-queries in QueryOptimizer.optimize(). Please also refer to this comment: https://issues.apache.org/jira/browse/PHOENIX-4585?focusedCommentId=16367616&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16367616
16. **summary:** Move join query optimization out from QueryCompiler into QueryOptimizer
    description: Currently we do optimization for join queries inside QueryCompiler, which makes the APIs and code logic confusing, so we need to move join optimization logic into QueryOptimizer. Similarly, but probably with a different approach, we need to optimize UNION ALL queries and derived table sub-queries in QueryOptimizer.optimize(). Please also refer to this comment: https://issues.apache.org/jira/browse/PHOENIX-4585?focusedCommentId=16367616&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16367616
    label: code-design
17. **summary:** Move join query optimization out from QueryCompiler into QueryOptimizer
    description: Currently we do optimization for join queries inside QueryCompiler, which makes the APIs and code logic confusing, so we need to move join optimization logic into QueryOptimizer. Similarly, but probably with a different approach, we need to optimize UNION ALL queries and derived table sub-queries in QueryOptimizer.optimize(). Please also refer to this comment: https://issues.apache.org/jira/browse/PHOENIX-4585?focusedCommentId=16367616&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16367616

jira_issues_comments:

1. **body:** Could you please take a look at this patch, [~jamestaylor]? Now {{PhoenixStatement.compilePlan()}} will only compile and not optimize at all, and only by calling {{PhoenixStatement.optimizePlan()}} or {{QueryOptimizer.optimize()}} can we get the optimized (index) plan. So from interface level, "optimize" behaves exactly the same for multiple-table queries (e.g., joins, sub-queries, union-all) as it does for single-table queries; while under the hood, we have {{QueryOptimizer.getApplicablePlans()}} work like: 1) *Delegate* single-table query to the original getApplicablePlans() call. 2) *Rewrite* table nodes: * For a join query, we compose an independent single-table query for each join table node, and use these queries find the optimal plan for

each join table node. Later on, we use this information to rewrite the whole join query, replacing data tables with index tables and columns accordingly wherever necessary. * For non-correlated sub-queries, for which we can't de-correlate (i.e., convert into join queries), we do the same thing as for join queries. Only that we will replace the sub-query with dummy values, so that columns in WHERE conditions can be leveraged in optimizations. * Other multiple-table queries, like UNION ALL, do not need table node rewrite. Proceed to the next step. 3) *Recompile* nested sub-queries (derived tables): * Re-compile the rewritten query with option "optimizeSubquery" = *true*. * Re-compile is necessary even if the query has not changed in step 2), because UNION ALL sub-selects and sub-queries in a sub-query node can only get a chance to be optimized in {{QueryCompiler.compileSubquery()}} with option "optimizeSubquery" turned on. Thus, joins and sub-queries nested in deeper levels can be optimized recursively. Challenges and potential improvements: # *Cost-based optimization*: find a global optimal plan for multiple-table queries. Right now we stop at the best plan for each individual join table node and do not consider that an alternative plan might be slightly more expensive in the individual plan but can make the whole query cost lower. The downside of this approach is that the search space for the optimal plan explodes as the number of table involved in a query goes up. # *Refine sub-query optimization process*. It would be optimal to find replacement plans for nested sub-queries all at once rather than call {{QueryCompiler.com.compile()}} with an "optimizeSubquery" flag. It wouldn't be necessarily more efficient than it is now, but it would be a code improvement, removing some co-dependence. I had tried this more radical solution but later found out that it was rather impossible to achieve due to the current lack of intermediate representation in the compiler. Tests: Added a few "testQueryPlanSourceRefsInXXX" tests to demonstrate that: * Before calling optimizePlan(), the table-refs are all data tables. * After calling optimizePlan(), the optimization has taken effect in all tables and all nested sub-queries. * It resolves PHOENIX-4617.
label: code-design

2. **body:** Thanks for the patch, [~maryannxue]. Looks really good - a nice improvement. A few questions: - Why is this change required? {code} *--- a/phoenix-core/src/it/java/org/apache/phoenix/end2end/join/BaseJoinIT.java +++ b/phoenix-core/src/it/java/org/apache/phoenix/end2end/join/BaseJoinIT.java @@ -34,6 +34,7 @@ import java.util.regex.Pattern; import org.apache.phoenix.cache.ServerCacheClient; import org.apache.phoenix.end2end.ParallelStatsDisabledIT; +import org.apache.phoenix.query.QueryServices; import org.apache.phoenix.util.PropertiesUtil; import org.apache.phoenix.util.SchemaUtil; import org.apache.phoenix.util.StringUtil; @@ -456,6 +457,7 @@ public abstract class BaseJoinIT extends ParallelStatsDisabledIT { protected Connection getConnection() throws SQLException { Properties props = PropertiesUtil.deepCopy(TEST_PROPERTIES); props.put(ServerCacheClient.HASH_JOIN_SERVER_CACHE_RESEND_PER_SERVER, "true"); + props.put(QueryServices.FORCE_ROW_KEY_ORDER_ATTRIB, "true"); return DriverManager.getConnection(getUrl(), props); } {code} - Minor nit, how about naming this class something like GenSubqueryParamValuesRewriter? {code} +public class SubqueryDummyValueRewriter extends ParseNodeRewriter { {code} - Do all the unit tests pass?
label: code-design
3. **body:** One more question: - Do you think it'd be a bit cleaner to have a QueryPlan.optimize() or getApplicablePlans() method instead of needing this instanceof check? {code} private List<QueryPlan> getApplicablePlans(QueryPlan dataPlan, PhoenixStatement statement, List<? extends PDatum> targetColumns, ParallelIteratorFactory parallelIteratorFactory, boolean stopAtBestPlan) throws SQLException { + if (!useIndexes) { + return Collections.singletonList(dataPlan); + } + + if (dataPlan instanceof BaseQueryPlan) { + return getApplicablePlans((BaseQueryPlan) dataPlan, statement, targetColumns, parallelIteratorFactory, stopAtBestPlan); + } {code}
label: code-design
4. **body:** Thank you very much for the review, [~jamestaylor]! # The change in the join tests was needed because I ran into a random failure in {{testJoinWithOffset()}} which I suspect was due to unstable result order since the test case did not specify an ORDER BY. # I thought it would be better to keep the optimization logic all in one place so it would be easier to maintain and extend later on. One way is to use the QueryPlanVisitor here to get rid of "instanceof" check while still keeping everything in QueryOptimizer. What do you think?
label: code-design
5. **body:** bq. I thought it would be better to keep the optimization logic all in one place so it would be easier to maintain and extend later on. One way is to use the QueryPlanVisitor here to get rid of "instanceof" check while still keeping everything in QueryOptimizer. What do you think? The visitor would be an improvement over an instanceof check. I suppose having a method to optimize a query plan becomes

more necessary if the logic is very different across the different query plans. I'm fine with whatever you think is best.
label: code-design

6. **body:** As one of the potential improvements I mentioned earlier, we might want to find a global optimal for some query plans, so I think it'd be good to keep the optimization logic in one place and independent of QueryPlan for the long-term goal. At this point, we only have three different situations to handle for all kinds of QueryPlan: # BaseQueryPlan # Joins # All others Ultimately we'd like to make it only two branches: 1. BaseQueryPlan and 2. Non-BaseQueryPlan. Although we do have to separate BaseQueryPlan from other kinds of QueryPlan, yet I don't think it's worth using a visitor either. Shall I just push this in as it is now and figure out what we should do as we expand the optimization logic?
label: code-design
7. Instead of this: {code} + if (dataPlan instanceof BaseQueryPlan) { + return getApplicablePlans((BaseQueryPlan) dataPlan, statement, targetColumns, parallelIteratorFactory, stopAtBestPlan); + } {code} can you do this? {code} if (dataPlan.getSourceRefs().size() == 1) { return getApplicablePlans(dataPlan, statement, targetColumns, parallelIteratorFactory, stopAtBestPlan); } {code}
8. Ping [~maryannxue]?
9. I tried it, [~jamestaylor], but it did not work. For derived-table queries like: {{select a, b from (select a, b from t1 order by b limit 50) where c = 'PPL'}} The query plan will be like ClientScanPlan(filter=" c = 'PPL' ", delegate=ScanPlan(table="t1", orderBy="b", limit=50)) So {{getSourceRefs}} will be of size 1 containing "t1", while the optimization for this query should go down the sub-query optimization path.
10. Is the derived-table the only exception and if so can we test for that too in some way? Alternatively, we could have three base classes for QueryPlan: - BaseQueryPlan (or FinalQueryPlan or PhysicalQueryPlan or LeafQueryPlan?) - JoinQueryPlan - IntermediateQueryPlan (or LogicalQueryPlan or NonLeafQueryPlan or CompositeQueryPlan) Then just have an optimize method on QueryPlan with an implementation on these three classes. If that's problematic, then go ahead and commit it as you have it.
11. FAILURE: Integrated in Jenkins build PreCommit-PHOENIX-Build #1823 (See [https://builds.apache.org/job/PreCommit-PHOENIX-Build/1823/]) PHOENIX-4616 Move join query optimization out from QueryCompiler into (maryannxue: rev ab16e2a27eaac70a1e1142d47307559fbde4bd49) * (edit) phoenix-core/src/main/java/org/apache/phoenix/jdbc/PhoenixStatement.java * (edit) phoenix-core/src/main/java/org/apache/phoenix/optimize/QueryOptimizer.java * (edit) phoenix-core/src/main/java/org/apache/phoenix/compile/SubselectRewriter.java * (edit) phoenix-core/src/main/java/org/apache/phoenix/compile/JoinCompiler.java * (add) phoenix-core/src/main/java/org/apache/phoenix/optimize/GenSubqueryParamValuesRewriter.java * (edit) phoenix-core/src/test/java/org/apache/phoenix/compile/QueryCompilerTest.java * (edit) phoenix-core/src/main/java/org/apache/phoenix/compile/UpsertCompiler.java * (edit) phoenix-core/src/it/java/org/apache/phoenix/end2end/join/BaseJoinIT.java * (edit) phoenix-core/src/main/java/org/apache/phoenix/compile/QueryCompiler.java
12. FAILURE: Integrated in Jenkins build Phoenix-4.x-HBase-1.3 #79 (See [https://builds.apache.org/job/Phoenix-4.x-HBase-1.3/79/]) PHOENIX-4616 Move join query optimization out from QueryCompiler into (maryannxue: rev 781a9c09c31d1e3090886b77d18638dfee7b615d) * (add) phoenix-core/src/main/java/org/apache/phoenix/optimize/GenSubqueryParamValuesRewriter.java * (edit) phoenix-core/src/main/java/org/apache/phoenix/jdbc/PhoenixStatement.java * (edit) phoenix-core/src/main/java/org/apache/phoenix/compile/SubselectRewriter.java * (edit) phoenix-core/src/main/java/org/apache/phoenix/compile/JoinCompiler.java * (edit) phoenix-core/src/main/java/org/apache/phoenix/optimize/QueryOptimizer.java * (edit) phoenix-core/src/it/java/org/apache/phoenix/end2end/join/BaseJoinIT.java * (edit) phoenix-core/src/main/java/org/apache/phoenix/compile/QueryCompiler.java * (edit) phoenix-core/src/main/java/org/apache/phoenix/compile/UpsertCompiler.java * (edit) phoenix-core/src/test/java/org/apache/phoenix/compile/QueryCompilerTest.java
13. FAILURE: Integrated in Jenkins build Phoenix-4.x-HBase-0.98 #1846 (See [https://builds.apache.org/job/Phoenix-4.x-HBase-0.98/1846/]) PHOENIX-4616 Move join query optimization out from QueryCompiler into (maryannxue: rev ca3ea728c2413b90d44bcaa1ae13381b79f6e978) * (edit) phoenix-core/src/main/java/org/apache/phoenix/jdbc/PhoenixStatement.java * (edit) phoenix-core/src/main/java/org/apache/phoenix/compile/JoinCompiler.java * (edit) phoenix-core/src/it/java/org/apache/phoenix/end2end/join/BaseJoinIT.java * (edit) phoenix-

core/src/test/java/org/apache/phoenix/compile/QueryCompilerTest.java * (edit) phoenix-core/src/main/java/org/apache/phoenix/optimize/QueryOptimizer.java * (add) phoenix-core/src/main/java/org/apache/phoenix/optimize/GenSubqueryParamValuesRewriter.java * (edit) phoenix-core/src/main/java/org/apache/phoenix/compile/SubselectRewriter.java * (edit) phoenix-core/src/main/java/org/apache/phoenix/compile/QueryCompiler.java * (edit) phoenix-core/src/main/java/org/apache/phoenix/compile/UpsertCompiler.java

14. Thank you, [~jamestaylor]! I'll keep the idea of refactoring the code in future improvements to query optimization. Anyway, whether to keep all optimization logic in one place or scattered into different classes, I think it would be wise to keep optimization of QueryPlan independent of QueryPlan classes' implementations, and to drive the optimization from outside.
15. Looks like your check-in broke CursorWithRowValueConstructorIT. It just hangs now which is breaking our build. Would you mind taking a look, please, [~maryannxue]?
16. Sorry that I didn't notice the error. Looking at it right now.
17. [~jamestaylor], I fixed it on master branch and the jenkins build is now running as https://builds.apache.org/blue/organizations/jenkins/Phoenix-master/detail/Phoenix-master/1979/pipeline/. Somehow all the transaction table tests have failed due to "unable to discover transaction service". I had verified those tests locally and they had been fine. Is there anything else that might be wrong?
18. Should be ok now, but let's see for the next build. Thanks!
19. FAILURE: Integrated in Jenkins build Phoenix-4.x-HBase-0.98 #1848 (See [https://builds.apache.org/job/Phoenix-4.x-HBase-0.98/1848/]) PHOENIX-4616 Move join query optimization out from QueryCompiler into (maryannxue: rev b52f467d970b2682b7a70952956144c965c472ba) * (edit) phoenix-core/src/main/java/org/apache/phoenix/optimize/QueryOptimizer.java
20. ABORTED: Integrated in Jenkins build Phoenix-4.x-HBase-1.3 #82 (See [https://builds.apache.org/job/Phoenix-4.x-HBase-1.3/82/]) PHOENIX-4616 Move join query optimization out from QueryCompiler into (maryannxue: rev fb9626117ec5d4c498c989f38fa077ab4acaef18) * (edit) phoenix-core/src/main/java/org/apache/phoenix/optimize/QueryOptimizer.java
21. FAILURE: Integrated in Jenkins build PreCommit-PHOENIX-Build #1824 (See [https://builds.apache.org/job/PreCommit-PHOENIX-Build/1824/]) PHOENIX-4616 Move join query optimization out from QueryCompiler into (maryannxue: rev 49fca494bf9e13918db558e8276676e3dfda9d74) * (edit) phoenix-core/src/main/java/org/apache/phoenix/optimize/QueryOptimizer.java PHOENIX-4616 Move join query optimization out from QueryCompiler into (maryannxue: rev 0b1b219ef0e803d7ff254408c24b4bb67a5d88f9) * (edit) phoenix-core/src/main/java/org/apache/phoenix/optimize/QueryOptimizer.java
22. FAILURE: Integrated in Jenkins build PreCommit-PHOENIX-Build #1930 (See [https://builds.apache.org/job/PreCommit-PHOENIX-Build/1930/]) PHOENIX-4616 Move join query optimization out from QueryCompiler into (maryannxue: rev 3c1b3b547f88cc2860b16658a0babe3e05d34c8e) * (edit) phoenix-core/src/main/java/org/apache/phoenix/compile/SubselectRewriter.java * (edit) phoenix-core/src/main/java/org/apache/phoenix/optimize/QueryOptimizer.java * (edit) phoenix-core/src/main/java/org/apache/phoenix/compile/QueryCompiler.java * (edit) phoenix-core/src/main/java/org/apache/phoenix/jdbc/PhoenixStatement.java * (edit) phoenix-core/src/main/java/org/apache/phoenix/compile/JoinCompiler.java * (add) phoenix-core/src/main/java/org/apache/phoenix/optimize/GenSubqueryParamValuesRewriter.java * (edit) phoenix-core/src/main/java/org/apache/phoenix/compile/UpsertCompiler.java * (edit) phoenix-core/src/it/java/org/apache/phoenix/end2end/join/BaseJoinIT.java * (edit) phoenix-core/src/test/java/org/apache/phoenix/compile/QueryCompilerTest.java PHOENIX-4616 Move join query optimization out from QueryCompiler into (maryannxue: rev 6521c87a03c001908f0d4fabf0f968e72c2d0a89) * (edit) phoenix-core/src/main/java/org/apache/phoenix/optimize/QueryOptimizer.java