Item 52
git_comments:

git_commits:

1. **summary:** [CALCITE-296] Re-organize package structure;
   message: [CALCITE-296] Re-organize package structure; [CALCITE-419] Naming convention for planner rules This change only renames files (and deletes some obsolete files). There are no content changes.
   label: code-design

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Re-organize package structure
   description: As we move code to the `org.apache.optiq` namespace, there is an opportunity to re-organize the package structure of the code. Proposed new structure is in a spreadsheet. ---------- Imported from GitHub ---------- Url: https://github.com/julianhyde/optiq/issues/296 Created by: [julianhyde|https://github.com/julianhyde] Labels: Created at: Thu May 29 21:09:56 CEST 2014 State: open

jira_issues_comments:

1. [Date: Thu May 29 22:13:18 CEST 2014, Author: [vlsi|https://github.com/vlsi]] Can we split `JavaRules` to separate standalone classes?
2. [Date: Thu May 29 22:23:01 CEST 2014, Author: [jacques-n|https://github.com/jacques-n]] What do you think of pulling all the impl packages out of core?
3. **body:** [Date: Thu May 29 23:01:16 CEST 2014, Author: [julianhyde|https://github.com/julianhyde]] @vlsi That makes sense. For what it's worth, here are all of the (non-generated) java files over 1,000 lines. We could consider splitting them up. ``` lines words chars file ======= ======= ======= == 1013 2972 30152 ./avatica/src/main/java/net/hydromatic/avatica/AvaticaResultSet.java 1018 3839 33057 ./core/src/main/java/org/eigenbase/rex/RexProgramBuilder.java 1040 3887 35947 ./core/src/main/java/org/eigenbase/relopt/SubstitutionVisitor.java 1043 2947 34398 ./core/src/main/java/org/eigenbase/sql2rel/StandardConvertletTable.java 1080 3224 34194 ./core/src/test/java/net/hydromatic/optiq/test/OptiqAssert.java 1098 3779 32867 ./core/src/main/java/org/eigenbase/rex/RexUtil.java 1126 3299 29610 ./core/src/main/java/net/hydromatic/optiq/runtime/AbstractCursor.java 1133 3109 30997 ./core/src/main/java/org/eigenbase/sql/pretty/SqlPrettyWriter.java 1199 4171 38151 ./core/src/test/java/org/eigenbase/sql/test/SqlAdvisorTest.java 1216 3993 35307 ./core/src/main/java/org/eigenbase/sql/type/SqlTypeUtil.java 1264 3860 39241 ./core/src/main/java/org/eigenbase/sql/SqlIntervalQualifier.java 1282 3688 41829 ./core/src/test/java/org/eigenbase/util/UtilTest.java 1302 4323 42896 ./core/src/main/java/org/eigenbase/rel/rules/ReduceDecimalsRule.java 1325 4274 41340 ./core/src/main/java/org/eigenbase/rex/RexBuilder.java 1364 4227 40318 ./core/src/test/java/org/eigenbase/test/concurrent/ConcurrentTestCommandGenerator.java 1382 5999 43647 ./core/src/main/java/org/eigenbase/util/RhBase64.java 1393 4271 49435 ./core/src/main/java/net/hydromatic/optiq/rules/java/RexImpTable.java 1473 4475 38653

./core/src/main/java/org/eigenbase/util/mapping/Mappings.java 1478 4113 45610
./core/src/main/java/org/eigenbase/sql/fun/SqlStdOperatorTable.java 1620 6601 44872
./core/src/main/java/net/hydromatic/optiq/runtime/SqlFunctions.java 1810 6397 57976
./core/src/main/java/org/eigenbase/relopt/volcano/VolcanoPlanner.java 2080 8116 61123
./core/src/main/java/org/eigenbase/util/Util.java 2105 6645 64425
./core/src/test/java/org/eigenbase/test/concurrent/ConcurrentTestCommandScript.java 2481 7503 85442
./core/src/main/java/org/eigenbase/sql2rel/RelDecorrelator.java 2619 8737 85995
./core/src/main/java/org/eigenbase/relopt/RelOptUtil.java 2892 7413 103702
./core/src/main/java/net/hydromatic/optiq/rules/java/JavaRules.java 4159 13175 137905
./core/src/main/java/org/eigenbase/sql/validate/SqlValidatorImpl.java 4885 15490 160008
./core/src/main/java/org/eigenbase/sql2rel/SqlToRelConverter.java 5038 14588 162342
./core/src/test/java/org/eigenbase/sql/test/SqlOperatorBaseTest.java 5235 21069 234394
./core/src/test/java/net/hydromatic/optiq/test/JdbcTest.java 5733 22120 188483
./core/src/test/java/org/eigenbase/sql/parser/SqlParserTest.java 6658 29294 252727
./core/src/test/java/org/eigenbase/test/SqlValidatorTest.java ``` Do we agree that it makes sense to have rels and rules in the same package for each adapter? (An adapter is, loosely, a calling convention, such as enumerable or mongodb.) I'd make an exception for the large number of logical rules. They deserve their own package. @jacques-n We're talking package structure here, which is orthogonal (I hope) to modules. But let's discuss modules anyway, because that gets us talking about dependencies. What do you think should remain in core? We could move out impl/clone and impl/jdbc, but impl/enumerable (formerly impl/java) would be tricky. Other adapters - jdbc, splunk and mongodb - depend on EnumerableConvention and EnumerableRel. And other code in core depends on code-generation - Prepare, ImplementableFunction, RexExecutorImpl. So, without enumerable, core can't execute SQL queries or provide JDBC. So we'd have to move out a lot of the system tests too. Also note that core depends on linq4j and avatica.
label: code-design

4. [Date: Fri May 30 23:53:38 CEST 2014, Author: [julianhyde|https://github.com/julianhyde]] I also propose we rename the `XxxRelBase` classes, and get rid of the `Rel` suffix. * `ProjectRelBase` to `Project`, `ProjectRel` to `LogicalProject` * `AggregateRelBase` to `Aggregate`, `AggregateRel` to `LogicalAggregate` * ditto `OneRow`, `Values`, `Join`, `Minus`, `TableFunction`, `TableModification`, `Union`, `Window`, `Calc`, `Intersect` * `SortRel` to `LogicalSort`, add `Sort` (there was never a `SortRelBase`) * `TableAccessRelBase` to `TableScanRel`, `TableAccessRel` to `LogicalTableScan` (note that "table access" becomes "scan") * `SingleRel` to `SingleRelNode` (no need for a `LogicalSingleRel`) * `EmptyRel` to `Empty` (no need to a `LogicalEmpty`) * `CollectRel` to `Collect` (no need for a `LogicalCollect`)
5. **body:** [Date: Sat May 31 00:15:13 CEST 2014, Author: [jacques-n|https://github.com/jacques-n]] The impl rip out is scary but I feel like it would ultimately create stronger core. We could also be much clearer about when we change core versus extended bits. Separating as modules also guarantees that we avoid implementation details leaking into the core. With regards to the base rel and logical rels, I think simplifying/cleaning names would be good. However, I would prefer that we distinguish between abstract classes and concrete classes by name. My preference is AbstractProject and Project but Project and LogicalProject is fine as well. My comment about naming is that I think it would be quite confusing if Project is abstract but Collect is not (especially since it is a final class).
label: code-design
6. **body:** [Date: Sat May 31 00:31:08 CEST 2014, Author: [julianhyde|https://github.com/julianhyde]] The reason I want to move to the shorter names is people are increasingly writing generic rules that use the base classes. If you are writing a rule that matches ProjectRelBase, and can therefore handle LogicalProject and DrillProject, you don't much care whether ProjectRelBase is abstract. In that sense, we are using ProjectRelBase as an interface. So I would choose the shorter name, Project, over AbstractProject, a name that is descriptive to some people but useless information to most.
label: code-design
7. [Date: Tue Jun 03 06:58:14 CEST 2014, Author: [vlsi|https://github.com/vlsi]] > you don't much care whether ProjectRelBase is abstract. In that sense, we are using ProjectRelBase as an interface. So I would choose the shorter name, Project, over AbstractProject I agree.
8. I took a stab just doing the simple renames in the spreadsheet, and got everything building and all the tests passing. I didn't do any of the other proposed refactoring aside from package renames. Also, I had to disable the import ordering checkstyle rule since there were zillions of failures; if the PRs are accepted, I can go through and get the imports ordered correctly. The pull requests are at: https://github.com/apache/incubator-optiq-linq4j/pull/1 https://github.com/apache/incubator-optiq/pull/4

https://github.com/apache/incubator-optiq-csv/pull/1 If you check them out to test, note that you need to do mvn install on each module in order for the subsequent modules to build.

9. We should get the green light from various stakeholders before doing this re-org. It may make some patches and branches out there impossible to merge. I'll send a message to the dev list. Tomorrow I'll see whether the branches I have (e.g. optiq-62b) can still be merged. I may have to ask you to re-do, when we have everything we need in master. How long did it take you? Sorry to be negative - but I am pleased to get this out of the blue contribution - thank you.
10. **body:** I can re-do it. It took a few hours, but the next time around would be faster: mostly just some scripting & tinkering.
    label: code-design
11. I pulled your changes and did some experiments. 1. linq4j and optiq built & installed just fine, and the optiq test suite passed. (Some errors in the spark adapter test, but minor.) 2. I notice that you moved net.hydromatic.linq4j -> org.apache.linq4j and net.hydromatic.avatica -> org.apache.avatica. I'm pretty sure apache aren't letting me have those namespaces. So those packages will need to be org.apache.optiq.linq4j and org.apache.optiq.avatica. 3. I don't want to do a parallel release of optiq-linq4j. In fact I don't ever want to release from the optiq-csv and optiq-linq4j repositories. So we need to copy the linq4j code into a new linq4j module of optiq, and remove the dependency on net.hydromatic.linq4j. Stretch goal, do the same for optiq-csv, which becomes a demo module. 4. Looking at https://github.com/mprudhom/incubator-optiq/commit/1a688e1a3b7fb7a08c12279e8a3bf5b2dd98c18c, it seems that the vast majority of changes are renames ('git mv'). Excellent. 5. There are a few deletes. What happened to core/src/main/java/net/hydromatic/optiq/impl/clone/package-info.java core/src/main/java/net/hydromatic/optiq/rules/java/package-info.java core/src/main/java/net/hydromatic/optiq/rules/package-info.java core/src/main/java/net/hydromatic/optiq/util/package-info.java 6. I tried 'git rebase -i master', where master had just one commit. Because you used 'git mv', conflicts were fairly limited. Mainly just in the import lists. However, I still can't see us doing large merges, e.g. https://github.com/julianhyde/optiq/compare/master...optiq-62b. I still think the plan should be to get the release ready in the "stable" branch (mainly dealing with red tape stuff like NOTICE files) and continue development on the "master" branch. The re-org will happen on "master", using the scripts you have developed, when we get the green light from the stakeholders. Then we should fix javadoc (checking using "mvn site") and fix imports (checking using "mvn validate" and checkstyle) before declaring the task complete. Do you agree? Can you please check in your script(s)? I think we can fix the bugs in the script then apply the script (and any manual changes) when we have the green light.
12. 1. I got a some weird errors in the Spark adapter tests too, having to do with HSQLDB driver handling. It seems unrelated to the package renaming, and I handled it by excluding the transitive dependency to an old version of the HSQL driver in the pom.xml. 2. Oops, I missed that rename. I'll get it next time. 3. Sounds good. I think we'll lose the git history for those migrated modules, right? 4. Yeah, it might even make it possible to merge in some simple outstanding branches after the reorganization. 5. I think those were mostly packages whose contents were merged into other existing packages. 6. Sounds good. I'll work on assembling a single master reorg script. I might just make it into a gist, since it is such a one-off process.
13. Agreed, on all points.
14. {quote}it seems that the vast majority of changes are renames ('git mv'). {quote} Git does not rely on {{git mv}} to detect renames. It just compares the content and understands that the file was moved no matter how do do the rename. {quote}3. Sounds good. I think we'll lose the git history for those migrated modules, right?{quote} Here's how we can keep history: {noformat} git clone https://github.com/julianhyde/optiq.git git subtree add --prefix=linq4j https://github.com/julianhyde/linq4j.git master git subtree add --prefix=example-csv https://github.com/julianhyde/optiq-csv.git master {noformat} Here's the result: https://github.com/vlsi/optiq-tmp/blame/retired/linq4j/src/main/java/net/hydromatic/linq4j/AbstractEnumerable.java This becomes unusual git tree (with multiple roots), however that is exactly what we want if we need history. Bonus point: I guess after-the-fact pull requests from linq4j and optiq-csv are possible as well, however that is not a priority. I often use commit history/blame in IDE to get familiar with the code. For instance, it enables you to link test-case with the required changes that made test work.
15. Great idea Vladimir!
16. Should we keep "eigenbase-properties" as an external dependency, or also pull it into the optiq project while we are at it? It's only 10 java classes; we could even just stick the files themselves into core without

making a separate module for them.

17. Agreed. 'git subtree' seems a good idea. Let's keep eigenbase-properties as an external dependency, since it has an independent purpose. (I might be able to obsolete it by doing code-generation, a la eigenbase-resource.)
18. OK, I've packaged it all together into a big gnarly bash script that checks out the optiq repository, merges in the external linq4j and cvs repositories, and fixes all the names as per the naming spreadsheet. Everything compiles and all the tests pass for me. Notably, the script does not: 1. Update the checkstyle rules or try to do any re-sorting of imports. Thus, mvn will fail unless you set "-Dcheckstyle.skip=true" 2. Update any of the README.md or other docs, and so they still reference older net.hydromatic classes 3. Modify any of the licensing headers Note that I've only tested it on MacOS 10.9, but I don't see any reason it wouldn't run on any UNIX. You can grab the script at: https://gist.github.com/mprudhom/181ec8251943dd57b0af
19. You can also see a preview of the code changes without running the script by cloning my test branch: git clone -b packagereorg https://github.com/mprudhom/incubator-optiq.git mvn -f incubator-optiq/pom.xml clean install -Dcheckstyle.skip=true
20. I think now is the time to start renaming packages. I have a few patches almost ready to merge into master, and then it's looking quiet for a few weeks. [~mprudhom] would you like to re-do the work that you did in July? Or would you like me to do it, running the scripts that you created. Let me know; I'm fine either way, but would like to get started in the next 2 or 3 days.
21. I probably won't have time to work on it in the coming week, so you go ahead. If you have any problems with the script, let me know; I expect it would only require a bit of tweaking.
22. Attach file describing before-after name mapping for CALCITE-296, CALCITE-419, CALCITE-465.
23. Fixed in http://git-wip-us.apache.org/repos/asf/incubator-calcite/commit/a611d645ad754844a39f3d98f5f814f13dbd9404 (move files) and http://git-wip-us.apache.org/repos/asf/incubator-calcite/commit/a0ba73cd2de76696b96a1cd828d2aa4d3ef9eb55 (fix up).
24. Closing now that 1.0.0-incubating has been released.