TAJO-1092

hyunsik

2014-10-02

Contents

Chapter 1

Root issue TAJO-1092

1.1 Summary

Improve the function system to allow other function implementation types

1.2 Description

In the current function system, each function implementation is a single Java class subclassed from org.apache.tajo.catalog.function.Function.
In this approach, there are many rooms for improvement. This approach always uses Datum as input and output values of functions, creating unnecessary objects. It does not likely to exploit given information included query statements; for example, some parameters are constants or variables.

In this issue, I propose the improvement to allow the function system to support other function implementation types. In addition, I propose three function implementation types:
- legacy Java class function provided by the current Tajo
- static method in Java class
- code generation by ASM

Later, we could expand this feature to allow Pig or Hive functions in Tajo.

1.3 Attachments

No attachments

1.4 Commits

1. Commit **d56737b** by **Hyunsik Choi** (2014-10-15): TAJO-1092: Improve the function system to allow other function implementation types.

   Closes #178

1.5 Comments

1. **githubbot:** GitHub user hyunsik opened a pull request:

https://github.com/apache/tajo/pull/178

TAJO-1092: Improve the function system to allow other function implementation types.

You can merge this pull request into a Git repository by running:

$ git pull https://github.com/hyunsik/tajo TAJO-1092

Alternatively you can review and apply these changes as the patch at:

https://github.com/apache/tajo/pull/178.patch

To close this pull request, make a commit to your master/trunk branch
with (at least) the following in the commit message:

This closes #178
---
commit 149b92afe0d5a2903ddd55b4e788482e3344477b
Author: Hyunsik Choi <hyunsik@apache.org>
Date: 2014-10-04T22:23:10Z

TAJO-1092: Improve the function system to allow other function implementation types.

commit 6df2f9f4d7c38f751b761f9dbdd224cc3c89b2ad
Author: Hyunsik Choi <hyunsik@apache.org>
Date: 2014-10-04T22:23:37Z

Merge branch 'master' of https://git-wip-us.apache.org/repos/asf/tajo into TAJO-1092

commit f6fe193152dc8a121def75b925bdde0e102a2cac
Author: Hyunsik Choi <hyunsik@apache.org>
Date: 2014-10-05T00:09:49Z

Refactored code generation for function binding.
* Moved some methods to TajoGeneratorAdaptor.
* Refactored method signatures of isMatched and isCompatible.
* Fixed unit test bugs of TestEvalCodeGenerator.

---

2. **githubbot:** Github user hyunsik commented on the pull request:

   https://github.com/apache/tajo/pull/178#issuecomment-57923814
   An example of static method in Java class is https://github.com/apache/tajo/pull/178/files#diff-f6daa76b2459470a9f3412131c0f726bR34.
   I designed the function annotation system to point Function Collection, which is a class including multiple static functions. For user-defined functions and built-in functions, just add function as the example. It is very easy and it enables Tajo to reuse existing functions.
   Besides, as you can see, SQL is based on three-valued logic (http://en.wikipedia.org/wiki/Three-valued_logic). So, each value can be nullable. Despite of boolean type, one boolean type value can be three values: TRUE, FALSE, and UNKNOWN (NULL in SQL). In the current function system, each function must deal with NULL value explicitly. Most of functions

usually return NULL if at least of one parameter is NULL. “‘Substr“‘ function is an example (https://github.com/apache/tajo/blob/master/tajo-core/src/main/java/org/apache/tajo/engine/function/string/S
It gives users burden, and it is easy for users to forget NULL handling when users implement user-defined functions.
In order to mitigate such a problem and to make function invocation more efficiently, I designed new function binder and new function definition approach to keep hints how a function handles NULL value.
The hints are described in function parameters in a function definition. You can specify the hints by using java primitive type or class primitive type as each parameter according to null handling way. For example:
This “‘pow“‘ function does not allow NULL values as input parameter. In this case, if at least one parameter is null, this function binder will automatically return NULL value without invoking this function. So, this function itself does not need to handle NULL value explicitly.
“‘
@ScalarFunction(name = "pow", returnType = FLOAT8, paramTypes = {FLOAT8, FLOAT8})
public static double pow(double x, double y) {
return Math.pow(x, y);
}
“‘
The following function definition allow NULL value as both input parameters. In this case, this function must handle NULL value explicitly.
“‘
@ScalarFunction(name = "pow", returnType = FLOAT8, paramTypes = {FLOAT8, FLOAT8})
public static Double pow(Double x, Double y) {
if (x == null || y == null) {
return null;
}
return Math.pow(x, y);
}
“‘
In addition, the function binder allows a mixed use of primitive types and class primitive types. When mixed definition is used, the function binder only allow class primitive types to handle NULL values explicitly.
Finally, the function binder is generated on the fly by java byte code generation technique, and it does not have any overheads even though the logic is very complex. Also, I’m expecting that this idea will remove significantly the overhead of Datum uses in the existing function system.

3. **githubbot:** Github user hyunsik commented on the pull request:

   https://github.com/apache/tajo/pull/178#issuecomment-57923825
   After this patch is committed, I’ll add a documentation about how making Tajo user-defined functions using the proposed design.

4. **githubbot:** Github user hyunsik commented on the pull request:

   https://github.com/apache/tajo/pull/178#issuecomment-58768370
   rebased.

5. **githubbot:** Github user jinossy commented on a diff in the pull request:

https://github.com/apache/tajo/pull/178#discussion_r18773411
--- Diff: tajo-core/src/main/java/org/apache/tajo/engine/function/math/MathFunctions.java
---
@@ -0,0 +1,46 @@
+/***
+ * Licensed to the Apache Software Foundation (ASF) under one
+ * or more contributor license agreements. See the NOTICE file
+ * distributed with this work for additional information
+ * regarding copyright ownership. The ASF licenses this file
+ * to you under the Apache License, Version 2.0 (the
+ * "License"); you may not use this file except in compliance
+ * with the License. You may obtain a copy of the License at
+ *
+ * http://www.apache.org/licenses/LICENSE-2.0
+ *
+ * Unless required by applicable law or agreed to in writing, software
+ * distributed under the License is distributed on an "AS IS" BASIS,
+ * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
+ * See the License for the specific language governing permissions and
+ * limitations under the License.
+ */
+
+package org.apache.tajo.engine.function.math;
+
+import org.apache.tajo.function.FunctionCollection;
+import org.apache.tajo.function.ScalarFunction;
+
+import static org.apache.tajo.common.TajoDataTypes.Type.FLOAT8;
+
+@FunctionCollection
+public class MathFunctions {
+
+ @ScalarFunction(name = "pi", returnType = FLOAT8)
+ public static double pi() {
+ return Math.PI;
+ }
+
+ @ScalarFunction(name = "pow", returnType = FLOAT8, paramTypes = {FLOAT8, FLOAT8})
+ public static Double pow(Double x, Double y) {
+ if (x == null || y == null) {
+ return null;
+ }
+ return Math.pow(x, y);
+ }
+
+// @ScalarFunction(name = "pow", returnType = FLOAT8, paramTypes = {FLOAT8, FLOAT8})
--- End diff --
Please remove the commented out line.

6. **githubbot:** Github user jinossy commented on the pull request:

https://github.com/apache/tajo/pull/178#issuecomment-58915658
Looks great to me!
In my opinion, you should check following example: public static bool myfunc(String x, int y); //nullable + primitive

7. **githubbot:** Github user hyunsik commented on the pull request:

   https://github.com/apache/tajo/pull/178#issuecomment-59063492
   The function support will be added in my next patch.

8. **githubbot:** Github user jinossy commented on the pull request:

   https://github.com/apache/tajo/pull/178#issuecomment-59147926
   This patch provide backward compatibility, so there is no issue.
   Here is my +1. Thank you for your contribution!

9. **githubbot:** Github user hyunsik commented on the pull request:

   https://github.com/apache/tajo/pull/178#issuecomment-59236781
   Thank you for your review. I'll commit it shortly.

10. **githubbot:** Github user asfgit closed the pull request at:

    https://github.com/apache/tajo/pull/178

11. **hudson:** SUCCESS: Integrated in Tajo-master-build #413 (See [https://builds.apache.org/job/Tajo-master-build/413/])
    TAJO-1092: Improve the function system to allow other function implementation types. (hyunsik: rev d56737b999ebb5ad822d32b336752f98cffb78f8)
    * tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/catalog/FunctionDesc.java
    * tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/catalog/function/Function.java
    * tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/DataTypeUtil.java
    * tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/function/ClassBaseInvocationDesc.java
    * tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/catalog/CatalogUtil.java
    * tajo-core/src/test/java/org/apache/tajo/engine/planner/TestExprAnnotator.java
    * tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/catalog/exception/NoSuchFunctionException
    * tajo-core/src/main/java/org/apache/tajo/engine/function/FunctionLoader.java
    * tajo-core/src/test/java/org/apache/tajo/client/TestTajoClient.java
    * tajo-core/src/main/java/org/apache/tajo/engine/function/builtin/AvgInt.java
    * tajo-core/src/main/java/org/apache/tajo/engine/function/GeneralFunction.java
    * tajo-core/src/test/java/org/apache/tajo/engine/eval/ExprTestBase.java
    * tajo-core/src/main/java/org/apache/tajo/util/JSPUtil.java
    * tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/function/Function.java
    * tajo-core/src/main/java/org/apache/tajo/engine/function/math/MathFunctions.java
    * tajo-core/src/main/java/org/apache/tajo/engine/codegen/CaseWhenEmitter.java
    * tajo-core/src/main/java/org/apache/tajo/engine/eval/AlgebraicUtil.java
    * tajo-core/src/test/java/org/apache/tajo/engine/planner/TestLogicalPlanner.java
    * tajo-catalog/tajo-catalog-common/src/main/proto/CatalogProtos.proto
    * tajo-core/src/test/java/org/apache/tajo/engine/planner/global/TestBroadcastJoinPlan.java
    * tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/catalog/json/FunctionAdapter.java

* tajo-core/src/main/java/org/apache/tajo/engine/codegen/EvalCodeGenerator.java
* tajo-core/src/main/java/org/apache/tajo/engine/codegen/VariablesPreBuilder.java
* tajo-catalog/tajo-catalog-common/src/test/java/org/apache/tajo/catalog/TestFunctionDesc.java
* tajo-core/src/main/java/org/apache/tajo/engine/planner/logical/join/JoinGraph.java
* tajo-core/src/main/java/org/apache/tajo/engine/function/AggFunction.java
* tajo-core/src/main/java/org/apache/tajo/engine/codegen/ScalarFunctionBindingEmitter.java
* tajo-core/src/main/java/org/apache/tajo/engine/utils/DataTypeUtil.java
* tajo-catalog/tajo-catalog-common/src/test/java/org/apache/tajo/catalog/TestCatalogUtil.java
* tajo-catalog/tajo-catalog-server/src/test/java/org/apache/tajo/catalog/TestCatalog.java
* tajo-core/src/main/java/org/apache/tajo/master/TajoMaster.java
* tajo-core/src/main/java/org/apache/tajo/engine/eval/WindowFunctionEval.java
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/function/FunctionSupplement.java
* tajo-core/src/test/java/org/apache/tajo/engine/eval/TestEvalTreeUtil.java
* tajo-core/src/test/java/org/apache/tajo/engine/function/TestMathFunctions.java
* tajo-core/src/main/java/org/apache/tajo/engine/optimizer/eval/rules/ConstantFolding.java
* tajo-core/src/main/java/org/apache/tajo/master/TajoMasterClientService.java
* tajo-core/src/main/java/org/apache/tajo/engine/codegen/TajoGeneratorAdapter.java
* tajo-core/src/main/java/org/apache/tajo/engine/plan/EvalTreeProtoDeserializer.java
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/function/FunctionSignature.java
* CHANGES
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/function/ScalarFunction.java
* tajo-core/src/main/java/org/apache/tajo/engine/eval/FunctionEval.java
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/function/FunctionUtil.java
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/function/StaticMethodInvocationDesc.java
* tajo-common/src/main/java/org/apache/tajo/util/ClassUtil.java
* tajo-core/src/test/java/org/apache/tajo/engine/planner/physical/TestPhysicalPlanner.java
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/catalog/json/CatalogGsonHelper.java
* tajo-core/src/main/java/org/apache/tajo/engine/planner/logical/join/GreedyHeuristicJoinOrderAlgorithm.java
* tajo-core/src/test/java/org/apache/tajo/master/TestGlobalPlanner.java
* tajo-core/src/main/java/org/apache/tajo/engine/json/CoreGsonHelper.java
* tajo-core/src/main/java/org/apache/tajo/engine/eval/BinaryEval.java
* tajo-core/src/main/java/org/apache/tajo/engine/planner/logical/join/JoinEdge.java
* tajo-core/src/main/java/org/apache/tajo/engine/planner/ExprAnnotator.java
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/function/FunctionInvocation.java
* tajo-core/src/main/java/org/apache/tajo/engine/planner/logical/TruncateTableNode.java
* tajo-core/src/test/resources/results/TestFunctionLoader/testFindScalarFunctions.result
* tajo-core/src/main/resources/webapps/admin/functions.jsp
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/function/FunctionCollection.java
* tajo-core/src/main/java/org/apache/tajo/engine/planner/TypeDeterminant.java
* tajo-core/src/test/java/org/apache/tajo/engine/planner/TestLogicalOptimizer.java
* tajo-core/src/main/java/org/apache/tajo/engine/function/math/Pow.java
* tajo-core/src/test/java/org/apache/tajo/LocalTajoTestingUtility.java
* tajo-core/src/main/java/org/apache/tajo/engine/codegen/LegacyFunctionBindingEmitter.java
* tajo-client/src/main/java/org/apache/tajo/cli/DescFunctionCommand.java
* tajo-core/src/test/java/org/apache/tajo/engine/function/TestFunctionLoader.java
* tajo-core/src/main/java/org/apache/tajo/util/ClassUtil.java
* tajo-common/src/main/java/org/apache/tajo/json/ClassNameSerializer.java
* tajo-catalog/tajo-catalog-server/src/main/java/org/apache/tajo/catalog/CatalogServer.java
* tajo-common/src/main/java/org/apache/tajo/util/TUtil.java

12. **hudson:** SUCCESS: Integrated in Tajo-master-CODEGEN-build #55 (See [https://builds.apache.org/job/Tajo-master-CODEGEN-build/55/])
    TAJO-1092: Improve the function system to allow other function implementation types.

(hyunsik: rev d56737b999ebb5ad822d32b336752f98cffb78f8)
* tajo-core/src/test/java/org/apache/tajo/engine/eval/TestEvalTreeUtil.java
* tajo-core/src/main/java/org/apache/tajo/engine/planner/logical/join/GreedyHeuristicJoinOrderAlgorithm.java
* tajo-core/src/test/java/org/apache/tajo/engine/eval/ExprTestBase.java
* tajo-core/src/main/java/org/apache/tajo/engine/planner/ExprAnnotator.java
* CHANGES
* tajo-core/src/main/java/org/apache/tajo/engine/planner/logical/join/JoinGraph.java
* tajo-core/src/main/java/org/apache/tajo/engine/function/builtin/AvgInt.java
* tajo-core/src/test/java/org/apache/tajo/engine/planner/TestExprAnnotator.java
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/catalog/json/CatalogGsonHelper.java
* tajo-catalog/tajo-catalog-common/src/test/java/org/apache/tajo/catalog/TestCatalogUtil.java
* tajo-core/src/main/java/org/apache/tajo/engine/eval/AlgebraicUtil.java
* tajo-core/src/main/java/org/apache/tajo/engine/eval/FunctionEval.java
* tajo-core/src/main/java/org/apache/tajo/master/TajoMasterClientService.java
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/catalog/CatalogUtil.java
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/catalog/FunctionDesc.java
* tajo-core/src/test/java/org/apache/tajo/engine/planner/TestLogicalPlanner.java
* tajo-core/src/main/java/org/apache/tajo/master/TajoMaster.java
* tajo-catalog/tajo-catalog-server/src/main/java/org/apache/tajo/catalog/CatalogServer.java
* tajo-common/src/main/java/org/apache/tajo/json/ClassNameSerializer.java
* tajo-catalog/tajo-catalog-common/src/main/proto/CatalogProtos.proto
* tajo-core/src/test/java/org/apache/tajo/engine/function/TestMathFunctions.java
* tajo-core/src/main/java/org/apache/tajo/engine/function/FunctionLoader.java
* tajo-core/src/test/java/org/apache/tajo/LocalTajoTestingUtility.java
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/function/StaticMethodInvocationDesc.java
* tajo-core/src/main/java/org/apache/tajo/engine/eval/BinaryEval.java
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/function/FunctionInvocation.java
* tajo-common/src/main/java/org/apache/tajo/util/TUtil.java
* tajo-core/src/main/java/org/apache/tajo/engine/function/AggFunction.java
* tajo-core/src/test/java/org/apache/tajo/engine/planner/physical/TestPhysicalPlanner.java
* tajo-client/src/main/java/org/apache/tajo/cli/DescFunctionCommand.java
* tajo-core/src/main/java/org/apache/tajo/engine/codegen/ScalarFunctionBindingEmitter.java
* tajo-core/src/main/java/org/apache/tajo/util/JSPUtil.java
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/function/FunctionSignature.java
* tajo-core/src/test/java/org/apache/tajo/client/TestTajoClient.java
* tajo-core/src/main/java/org/apache/tajo/engine/planner/logical/TruncateTableNode.java
* tajo-core/src/main/java/org/apache/tajo/engine/eval/WindowFunctionEval.java
* tajo-core/src/test/java/org/apache/tajo/engine/planner/TestLogicalOptimizer.java
* tajo-catalog/tajo-catalog-common/src/test/java/org/apache/tajo/catalog/TestFunctionDesc.java
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/function/ClassBaseInvocationDesc.java
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/function/FunctionSupplement.java
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/function/Function.java
* tajo-core/src/main/java/org/apache/tajo/engine/function/GeneralFunction.java
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/catalog/json/FunctionAdapter.java
* tajo-core/src/main/java/org/apache/tajo/engine/utils/DataTypeUtil.java
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/DataTypeUtil.java
* tajo-core/src/main/java/org/apache/tajo/engine/json/CoreGsonHelper.java
* tajo-core/src/main/java/org/apache/tajo/engine/codegen/EvalCodeGenerator.java
* tajo-core/src/test/java/org/apache/tajo/master/TestGlobalPlanner.java
* tajo-core/src/main/java/org/apache/tajo/util/ClassUtil.java
* tajo-core/src/main/java/org/apache/tajo/engine/codegen/VariablesPreBuilder.java
* tajo-core/src/test/resources/results/TestFunctionLoader/testFindScalarFunctions.result
* tajo-core/src/main/java/org/apache/tajo/engine/codegen/CaseWhenEmitter.java
* tajo-core/src/main/java/org/apache/tajo/engine/plan/EvalTreeProtoDeserializer.java

* tajo-core/src/main/resources/webapps/admin/functions.jsp
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/function/ScalarFunction.java
* tajo-core/src/main/java/org/apache/tajo/engine/planner/TypeDeterminant.java
* tajo-core/src/test/java/org/apache/tajo/engine/function/TestFunctionLoader.java
* tajo-core/src/main/java/org/apache/tajo/engine/codegen/LegacyFunctionBindingEmitter.java
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/catalog/function/Function.java
* tajo-common/src/main/java/org/apache/tajo/util/ClassUtil.java
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/function/FunctionUtil.java
* tajo-core/src/main/java/org/apache/tajo/engine/function/math/Pow.java
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/catalog/exception/NoSuchFunctionException
* tajo-core/src/main/java/org/apache/tajo/engine/codegen/TajoGeneratorAdapter.java
* tajo-core/src/test/java/org/apache/tajo/engine/planner/global/TestBroadcastJoinPlan.java
* tajo-catalog/tajo-catalog-server/src/test/java/org/apache/tajo/catalog/TestCatalog.java
* tajo-core/src/main/java/org/apache/tajo/engine/function/math/MathFunctions.java
* tajo-core/src/main/java/org/apache/tajo/engine/planner/logical/join/JoinEdge.java
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/function/FunctionCollection.java
* tajo-core/src/main/java/org/apache/tajo/engine/optimizer/eval/rules/ConstantFolding.java

13. **hyunsik:** Just committed this patch to master and block_iteration branches.

14. **hudson:** SUCCESS: Integrated in Tajo-block_iteration-branch-build #16 (See [https://builds.apache.org/job/Tajo
block_iteration-branch-build/16/])
TAJO-1092: Improve the function system to allow other function implementation types.
(hyunsik: rev 05ca386c45adbb551ebc3af22592ea0dd38b1ea4)
* tajo-common/src/main/java/org/apache/tajo/util/TUtil.java
* tajo-core/src/main/java/org/apache/tajo/engine/codegen/VariablesBuilder.java
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/function/ClassBaseInvocationDesc.java
* tajo-core/src/main/java/org/apache/tajo/master/TajoMasterClientService.java
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/DataTypeUtil.java
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/catalog/FunctionDesc.java
* tajo-core/src/main/java/org/apache/tajo/engine/planner/logical/join/GreedyHeuristicJoinOrderAlgorithm.java
* tajo-core/src/main/java/org/apache/tajo/engine/planner/logical/TruncateTableNode.java
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/catalog/exception/NoSuchFunctionException
* tajo-common/src/main/java/org/apache/tajo/json/ClassNameSerializer.java
* tajo-catalog/tajo-catalog-common/src/test/java/org/apache/tajo/catalog/TestCatalogUtil.java
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/catalog/CatalogUtil.java
* tajo-core/src/test/resources/results/TestFunctionLoader/testFindScalarFunctions.result
* tajo-core/src/main/resources/webapps/admin/functions.jsp
* tajo-core/src/test/java/org/apache/tajo/engine/function/TestMathFunctions.java
* tajo-core/src/test/java/org/apache/tajo/engine/planner/TestLogicalPlanner.java
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/function/StaticMethodInvocationDesc.java
* tajo-core/src/main/java/org/apache/tajo/engine/optimizer/eval/rules/ConstantFolding.java
* tajo-core/src/test/java/org/apache/tajo/LocalTajoTestingUtility.java
* tajo-core/src/main/java/org/apache/tajo/engine/function/FunctionLoader.java
* tajo-core/src/test/java/org/apache/tajo/engine/planner/TestLogicalOptimizer.java
* tajo-common/src/main/java/org/apache/tajo/util/ClassUtil.java
* tajo-core/src/main/java/org/apache/tajo/engine/plan/EvalTreeProtoDeserializer.java
* tajo-core/src/test/java/org/apache/tajo/engine/planner/physical/TestPhysicalPlanner.java
* tajo-core/src/main/java/org/apache/tajo/engine/planner/ExprAnnotator.java
* tajo-core/src/main/java/org/apache/tajo/util/JSPUtil.java
* tajo-core/src/main/java/org/apache/tajo/engine/utils/DataTypeUtil.java
* tajo-core/src/main/java/org/apache/tajo/engine/planner/logical/join/JoinEdge.java
* tajo-catalog/tajo-catalog-server/src/test/java/org/apache/tajo/catalog/TestCatalog.java
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/function/Function.java

* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/function/FunctionInvocation.java
* tajo-core/src/main/java/org/apache/tajo/util/ClassUtil.java
* tajo-core/src/main/java/org/apache/tajo/engine/codegen/EvalCodeGenerator.java
* tajo-core/src/main/java/org/apache/tajo/engine/function/math/Pow.java
* tajo-core/src/main/java/org/apache/tajo/engine/json/CoreGsonHelper.java
* tajo-core/src/main/java/org/apache/tajo/engine/function/AggFunction.java
* tajo-core/src/main/java/org/apache/tajo/engine/eval/WindowFunctionEval.java
* tajo-core/src/test/java/org/apache/tajo/client/TestTajoClient.java
* tajo-core/src/test/java/org/apache/tajo/master/TestGlobalPlanner.java
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/function/FunctionUtil.java
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/catalog/function/Function.java
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/function/FunctionSignature.java
* tajo-core/src/main/java/org/apache/tajo/engine/function/GeneralFunction.java
* tajo-core/src/test/java/org/apache/tajo/engine/function/TestFunctionLoader.java
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/function/FunctionCollection.java
* tajo-core/src/main/java/org/apache/tajo/engine/codegen/LegacyFunctionBindingEmitter.java
* tajo-catalog/tajo-catalog-common/src/test/java/org/apache/tajo/catalog/TestFunctionDesc.java
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/function/FunctionSupplement.java
* tajo-core/src/main/java/org/apache/tajo/engine/codegen/TajoGeneratorAdapter.java
* tajo-core/src/main/java/org/apache/tajo/engine/function/math/MathFunctions.java
* tajo-core/src/main/java/org/apache/tajo/master/TajoMaster.java
* tajo-core/src/main/java/org/apache/tajo/engine/eval/BinaryEval.java
* tajo-catalog/tajo-catalog-common/src/main/proto/CatalogProtos.proto
* tajo-core/src/main/java/org/apache/tajo/engine/codegen/ScalarFunctionBindingEmitter.java
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/function/ScalarFunction.java
* tajo-core/src/test/java/org/apache/tajo/engine/eval/ExprTestBase.java
* tajo-core/src/main/java/org/apache/tajo/engine/codegen/CaseWhenEmitter.java
* CHANGES
* tajo-core/src/test/java/org/apache/tajo/engine/planner/TestExprAnnotator.java
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/catalog/json/FunctionAdapter.java
* tajo-core/src/main/java/org/apache/tajo/engine/planner/TypeDeterminant.java
* tajo-core/src/main/java/org/apache/tajo/engine/planner/logical/join/JoinGraph.java
* tajo-core/src/main/java/org/apache/tajo/engine/function/builtin/AvgInt.java
* tajo-core/src/test/java/org/apache/tajo/engine/planner/global/TestBroadcastJoinPlan.java
* tajo-catalog/tajo-catalog-server/src/main/java/org/apache/tajo/catalog/CatalogServer.java
* tajo-client/src/main/java/org/apache/tajo/cli/DescFunctionCommand.java
* tajo-core/src/main/java/org/apache/tajo/engine/eval/AlgebraicUtil.java
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/catalog/json/CatalogGsonHelper.java
* tajo-core/src/main/java/org/apache/tajo/engine/eval/FunctionEval.java
* tajo-core/src/test/java/org/apache/tajo/engine/eval/TestEvalTreeUtil.java

1.6 Pull requests

1.6.1 Pull request 178

Title: TAJO-1092: Improve the function system to allow other function implementation types.
Author: hyunsik
Date: 2014-10-05
Status: closed
Comments:

1. **hyunsik** (2014-10-05): An example of static method in Java class is https://github.com/apache/tajo/pull/178/files#
f6daa76b2459470a9f3412131c0f726bR34.

I designed the function annotation system to point Function Collection, which is a class including multiple static functions. For user-defined functions and built-in functions, just add function as the example. It is very easy and it enables Tajo to reuse existing functions.

Besides, as you can see, SQL is based on three-valued logic (http://en.wikipedia.org/wiki/Three-valued_logic). So, each value can be nullable. Despite of boolean type, one boolean type value can be three values: TRUE, FALSE, and UNKNOWN (NULL in SQL). In the current function system, each function must deal with NULL value explicitly. Most of functions usually return NULL if at least of one parameter is NULL. 'Substr' function is an example (https://github.com/apache/tajo/blob/master/tajo-core/src/main/java/org/apache/tajo/engine/function/string/S
It gives users burden, and it is easy for users to forget NULL handling when users implement user-defined functions.

In order to mitigate such a problem and to make function invocation more efficiently, I designed new function binder and new function definition approach to keep hints how a function handles NULL value.

The hints are described in function parameters in a function definition. You can specify the hints by using java primitive type or class primitive type as each parameter according to null handling way.

For example:

This 'pow' function does not allow NULL values as input parameter. In this case, if at least one parameter is null, this function binder will automatically return NULL value without invoking this function. So, this function itself does not need to handle NULL value explicitly.

```
@ScalarFunction(name = "pow", returnType = FLOAT8, paramTypes = {FLOAT8, FLOAT8})
public static double pow(double x, double y) {
return Math.pow(x, y);
}
```

The following function definition allow NULL value as both input parameters. In this case, this function must handle NULL value explicitly.

```
@ScalarFunction(name = "pow", returnType = FLOAT8, paramTypes = {FLOAT8, FLOAT8})
public static Double pow(Double x, Double y) {
if (x == null || y == null) {
return null;
}
return Math.pow(x, y);
}
```

In addition, the function binder allows a mixed use of primitive types and class primitive types. When mixed definition is used, the function binder only allow class primitive types to handle NULL values explicitly.

Finally, the function binder is generated on the fly by java byte code generation technique, and it does not have any overheads even though the logic is very complex. Also, I'm ex-

pecting that this idea will remove significantly the overhead of Datum uses in the existing function system.

2. **hyunsik** (2014-10-05): After this patch is committed, I'll add a documentation about how making Tajo user-defined functions using the proposed design.

3. **hyunsik** (2014-10-11): rebased.

4. **jinossy** (2014-10-13): Looks great to me!
   In my opinion, you should check following example:
   public static bool myfunc(String x, int y); //nullable + primitive

5. **hyunsik** (2014-10-14): The function support will be added in my next patch. Thank you for your review.

6. **jinossy** (2014-10-15): This patch provide backward compatibility, so there is no issue.
   Here is my +1. Thank you for your contribution!

7. **hyunsik** (2014-10-15): Thank you for your review. I'll commit it shortly.

Chapter 2

Connected issue TAJO-1057

2.1 Summary

Function and UDF mechanism should not use Datum as parameter and return values

2.2 Description

The current function and UDF mechanism uses Datum as function arguments and return values. For more efficient processing, we should not use Datum objects and instead we should use Java primitive values.

2.3 Attachments

No attachments

2.4 Commits

No related commits

2.5 Comments

No comments

2.6 Pull requests

No pull requests

Chapter 3

Connected issue TAJO-1058

3.1 Summary

Implement off-heap hash table for hash aggregation

3.2 Description

See the title.

Currently, we use HashMap for hash aggregation. It causes GC overheads when the number of distinct hash keys is over tens of millions. We need to implement an alternative to the current implementation. This implementation should consider block iteration addressed in TAJO-1041.

3.3 Attachments

No attachments

3.4 Commits

No related commits

3.5 Comments

1. **hyunsik:** Off-heap hash table should keep aggregation contexts into off-heap memory space. For this, it requires a different kind function implementation. TAJO-1092 will resolve this problem.
2. **atris:** As a curious onlooker, I understand that this problem is solved with 1092 closing. Is that correct,please?

3.6 Pull requests

No pull requests