Item 192
git_comments:

git_commits:

1. **summary:** [FLINK-7194] [table] Add default implementations for type hints to UDAGG interface.
   message: [FLINK-7194] [table] Add default implementations for type hints to UDAGG interface. This closes #4379

github_issues:

github_issues_comments:

github_pulls:

1. **title:** [FLINK-7194] [table] Add methods for type hints to UDAGG interface.
   body: - [X] General - The pull request references the related JIRA issue ("[FLINK-XXX] Jira title text") - The pull request addresses only one issue - Each commit in the PR has a meaningful commit message (including the JIRA id) - [ ] Documentation - Documentation for UDAGGs will be provided by FLINK-6751 - [X] JavaDoc for public methods has been added - [X] Tests & Build - Functionality added by the pull request is covered by tests - `mvn clean verify` has been executed successfully locally or a Travis build has passed

github_pulls_comments:

1. Thanks for the review @sunjincheng121. I updated the PR.
2. Hi @fhueske Thanks for the update. The PR. looks good to me. -:) +1 to merge.
3. Loos good to me. +1 to merge
4. Merging

github_pulls_reviews:

1. `TypeInformation for the result ` -> `TypeInformation of the result `
2. ` Returns the TypeInformation for` -> ` Returns of TypeInformation for`
3. `getAccumulatorType()`->`getAccumulatorType`
4. **body:** Suggest to remove the useless java doc, something like: 1. line 35 `- getAccumulatorType.` 2. line 102 to line 112 `def getResultType: TypeInformation[_]`
   label: documentation
5. How about classOf[IntegralAvgAccumulator], Although classOf[T] is equivalent to the class literal T.class in Java. but I think it is more concise.What do you think?
6. Just a suggestion(I am not sure the suggestion is better or not): Extract common code for `getResultTypeOfAggregateFunction` and `getAccumulatorTypeOfAggregateFunction`. Something like following: ![image](https://user-images.githubusercontent.com/22488084/28444428-c85f8dc2-6def-11e7-996e-de80a728a6ce.png)
7. Good catch, thanks
8. I moved the TypeExtractor part into a separate method
9. Sounds good.

jira_issues:

1. **summary:** Add getResultType and getAccumulatorType to AggregateFunction
   description: FLINK-6725 and FLINK-6457 proposed to remove methods with default implementations such as {{getResultType()}}, {{toString()}}, or {{requiresOver()}} from the base classes of user-defined methods (UDF, UDTF, UDAGG) and instead offer them as contract methods which are dynamically In PR [#3993|https://github.com/apache/flink/pull/3993] I argued that these methods have a fixed signature (in contrast to the {{eval()}}, {{accumulate()}} and {{retract()}} methods) and should be kept in the classes. For users that don't need these methods, this doesn't make a difference because the methods are not abstract and have a default implementation. For users that need to override the methods it makes a difference, because they get IDE and compiler support when overriding them and the cannot get the signature wrong. Consequently, I propose to add {{getResultType()}} and {{getAccumulatorType()}} as

methods with default implementation to {{AggregateFunction}}. This will make the interface of {{AggregateFunction}} more consistent with {{ScalarFunction}} and {{TableFunction}}. What do you think [~shaoxuan], [~RuidongLi] and [~jark]?

jira_issues_comments:

1. **body:** [~fhueske]I agree to your suggestion, but the {{getResultType()}} in {{ScalarFunction}} is different from {{TableFunction}} and {{AggregateFunction}}, what about if we put a ban on overloading {{eval()}} in {{ScalarFunction}}? That will result in better consistence.
   label: code-design
2. In what sense is {{ScalarFunction.getResultType()}} different from {{TableFunction}} and {{AggregateFunction}} and how is {{ScalarFunction.eval()}} related to that? Thanks, Fabian
3. {{ScalarFunction.getResultType()}} has parameters while {{TableFunction}} and {{AggregateFunction}} does not, users can implement different {{ScalarFunction.eval()}} with different signatures, such as {{def eval(x: Int): Boolean}} or {{def eval(x: String): String}}, so the {{ScalarFunction.getResultType()}}' s return value is determined by parameters.
4. Oh I see what you mean. Thanks [~RuidongLi]. I would treat - adding {{getResultType()}} and {{getAccumulatorType()}} to {{AggregateFunction and - changing {{ScalarFunction}} to only support a single {{eval()}} method independently and rather open a new JIRA to propose changes on the {{ScalarFunction}}. Thanks, Fabian
5. [~fhueske], even if we change {{ScalarFunction.getResultType()}} to without parameters, we can also support multiple {{eval()}} method but with the same return type.
6. You are of course right [~jark]. Unless there are any objections, I would continue to open a PR to make {{getResultType()}} and {{getAccumulatorType()}} member methods of {{AggregateFunction}}. We can discuss changes on {{ScalarFunction}} in a separate JIRA.
7. [~fhueske], I am ok with your proposal for the changes to the {{AggregateFunction}}
8. +1 to do that
9. GitHub user fhueske opened a pull request: https://github.com/apache/flink/pull/4379 [FLINK-7194] [table] Add methods for type hints to UDAGG interface. - [X] General - The pull request references the related JIRA issue ("[FLINK-XXX] Jira title text") - The pull request addresses only one issue - Each commit in the PR has a meaningful commit message (including the JIRA id) - [ ] Documentation - Documentation for UDAGGs will be provided by FLINK-6751 - [X] JavaDoc for public methods has been added - [X] Tests & Build - Functionality added by the pull request is covered by tests - `mvn clean verify` has been executed successfully locally or a Travis build has passed You can merge this pull request into a Git repository by running: $ git pull https://github.com/fhueske/flink tableUDAGG Alternatively you can review and apply these changes as the patch at: https://github.com/apache/flink/pull/4379.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 #4379 ---- commit 22a56c0a2c7e4017b2c3bda56d07cdd6c5d39144 Author: Fabian Hueske <fhueske@apache.org> Date: 2017-07-20T13:09:06Z [FLINK-7194] [table] Add default implementations for type hints to UDAGG interface. ----
10. Github user sunjincheng121 commented on a diff in the pull request: https://github.com/apache/flink/pull/4379#discussion_r128658352 --- Diff: flink-libraries/flink-table/src/main/scala/org/apache/flink/table/functions/aggfunctions/SumWithRetractAggFunction.scala --- @@ -191,10 +191,10 @@ class DecimalSumWithRetractAggFunction acc.f1 = 0L } - def getAccumulatorType(): TypeInformation[_] = { + override def getAccumulatorType(): TypeInformation[DecimalSumWithRetractAccumulator] = { --- End diff -- `getAccumulatorType()`->`getAccumulatorType`
11. Github user sunjincheng121 commented on a diff in the pull request: https://github.com/apache/flink/pull/4379#discussion_r128662028 --- Diff: flink-libraries/flink-table/src/main/scala/org/apache/flink/table/functions/utils/UserDefinedFunctionUtils.scala --- @@ -314,7 +314,28 @@ object UserDefinedFunctionUtils { aggregateFunction: AggregateFunction[_, _], extractedType: TypeInformation[_] = null) : TypeInformation[_] = { - getParameterTypeOfAggregateFunction(aggregateFunction, "getResultType", 0, extractedType) + + val resultType = aggregateFunction.getResultType + if (resultType != null) { + resultType + } else if (extractedType != null) { + extractedType + } else { + try { + TypeExtractor + .createTypeInfo(aggregateFunction, + classOf[AggregateFunction[_, _]], + aggregateFunction.getClass, + 0) + .asInstanceOf[TypeInformation[_]] + } catch { + case ite: InvalidTypesException => + throw new TableException(+ s"Cannot infer generic type of ${aggregateFunction.getClass}. " + + s"You can

override AggregateFunction.getResultType() to specify the type.", + ite) + } + } --- End diff -- Just a suggestion(I am not sure the suggestion is better or not): Extract common code for `getResultTypeOfAggregateFunction` and `getAccumulatorTypeOfAggregateFunction`. Something like following: ![image](https://user-images.githubusercontent.com/22488084/28444428-c85f8dc2-6def-11e7-996e-de80a728a6ce.png)

12. Github user sunjincheng121 commented on a diff in the pull request: https://github.com/apache/flink/pull/4379#discussion_r128658053 --- Diff: flink-libraries/flink-table/src/main/scala/org/apache/flink/table/functions/AggregateFunction.scala --- @@ -136,8 +137,26 @@ abstract class AggregateFunction[T, ACC] extends UserDefinedFunction { */ def getValue(accumulator: ACC): T - /** - * whether this aggregate only used in OVER clause + /** + * Returns true if this AggregateFunction can only be applied in an OVER window. + * + * @return true if the AggregateFunction requires an OVER window, false otherwise. */ def requiresOver: Boolean = false + + /** + * Returns the TypeInformation for the result of the AggregateFunction. + * + * @return The TypeInformation of the result of the AggregateFunction or null if the result type + * should be automatically inferred. + */ + def getResultType: TypeInformation[T] = null + + /** + * Returns the TypeInformation for the accumulator of the AggregateFunction. --- End diff -- ` Returns the TypeInformation for` -> ` Returns of TypeInformation for`
13. Github user sunjincheng121 commented on a diff in the pull request: https://github.com/apache/flink/pull/4379#discussion_r128659642 --- Diff: flink-libraries/flink-table/src/main/scala/org/apache/flink/table/functions/AggregateFunction.scala --- @@ -17,6 +17,8 @@ */ package org.apache.flink.table.functions --- End diff -- Suggest to remove the useless java doc, something like: 1. line 35 `- getAccumulatorType.` 2. line 102 to line 112 `def getResultType: TypeInformation[_]`
14. Github user sunjincheng121 commented on a diff in the pull request: https://github.com/apache/flink/pull/4379#discussion_r128657888 --- Diff: flink-libraries/flink-table/src/main/scala/org/apache/flink/table/functions/AggregateFunction.scala --- @@ -136,8 +137,26 @@ abstract class AggregateFunction[T, ACC] extends UserDefinedFunction { */ def getValue(accumulator: ACC): T - /** - * whether this aggregate only used in OVER clause + /** + * Returns true if this AggregateFunction can only be applied in an OVER window. + * + * @return true if the AggregateFunction requires an OVER window, false otherwise. */ def requiresOver: Boolean = false + + /** + * Returns the TypeInformation for the result of the AggregateFunction. --- End diff -- `TypeInformation for the result ` -> `TypeInformation of the result `
15. Github user sunjincheng121 commented on a diff in the pull request: https://github.com/apache/flink/pull/4379#discussion_r128660377 --- Diff: flink-libraries/flink-table/src/main/scala/org/apache/flink/table/functions/aggfunctions/AvgAggFunction.scala --- @@ -80,11 +80,11 @@ abstract class IntegralAvgAggFunction[T] extends AggregateFunction[T, IntegralAv acc.f1 = 0L } - def getAccumulatorType: TypeInformation[_] = { + override def getAccumulatorType: TypeInformation[IntegralAvgAccumulator] = { new TupleTypeInfo(new IntegralAvgAccumulator().getClass, --- End diff -- How about classOf[IntegralAvgAccumulator], Although classOf[T] is equivalent to the class literal T.class in Java. but I think it is more concise.What do you think?
16. Github user fhueske commented on a diff in the pull request: https://github.com/apache/flink/pull/4379#discussion_r128728039 --- Diff: flink-libraries/flink-table/src/main/scala/org/apache/flink/table/functions/AggregateFunction.scala --- @@ -17,6 +17,8 @@ */ package org.apache.flink.table.functions --- End diff -- Good catch, thanks
17. Github user fhueske commented on a diff in the pull request: https://github.com/apache/flink/pull/4379#discussion_r128745444 --- Diff: flink-libraries/flink-table/src/main/scala/org/apache/flink/table/functions/utils/UserDefinedFunctionUtils.scala --- @@ -314,7 +314,28 @@ object UserDefinedFunctionUtils { aggregateFunction: AggregateFunction[_, _], extractedType: TypeInformation[_] = null) : TypeInformation[_] = { - getParameterTypeOfAggregateFunction(aggregateFunction, "getResultType", 0, extractedType) + + val resultType = aggregateFunction.getResultType + if (resultType != null) { + resultType + } else if (extractedType != null) { + extractedType + } else { + try { + TypeExtractor + .createTypeInfo(aggregateFunction, + classOf[AggregateFunction[_, _]], + aggregateFunction.getClass, + 0) + .asInstanceOf[TypeInformation[_]] + } catch { + case ite: InvalidTypesException => + throw new TableException(+ s"Cannot infer generic type of ${aggregateFunction.getClass}. " + + s"You can override AggregateFunction.getResultType() to specify the type.", + ite) + } + } --- End diff -- I moved the TypeExtractor part into a separate method

18. Github user fhueske commented on the issue: https://github.com/apache/flink/pull/4379 Thanks for the review @sunjincheng121. I updated the PR.
19. Github user sunjincheng121 commented on a diff in the pull request: https://github.com/apache/flink/pull/4379#discussion_r128773463 --- Diff: flink-libraries/flink-table/src/main/scala/org/apache/flink/table/functions/utils/UserDefinedFunctionUtils.scala --- @@ -314,7 +314,28 @@ object UserDefinedFunctionUtils { aggregateFunction: AggregateFunction[_, _], extractedType: TypeInformation[_] = null) : TypeInformation[_] = { - getParameterTypeOfAggregateFunction(aggregateFunction, "getResultType", 0, extractedType) + + val resultType = aggregateFunction.getResultType + if (resultType != null) { + resultType + } else if (extractedType != null) { + extractedType + } else { + try { + TypeExtractor + .createTypeInfo(aggregateFunction, + classOf[AggregateFunction[_, _]], + aggregateFunction.getClass, + 0) + .asInstanceOf[TypeInformation[_]] + } catch { + case ite: InvalidTypesException => + throw new TableException(+ s"Cannot infer generic type of ${aggregateFunction.getClass}. " + + s"You can override AggregateFunction.getResultType() to specify the type.", + ite) + } + } --- End diff -- Sounds good.
20. Github user sunjincheng121 commented on the issue: https://github.com/apache/flink/pull/4379 Hi @fhueske Thanks for the update. The PR. looks good to me. -:) + to merge.
21. Github user wuchong commented on the issue: https://github.com/apache/flink/pull/4379 Loos good to me. +1 to merge
22. Github user sunjincheng121 commented on the issue: https://github.com/apache/flink/pull/4379 Merging
23. Github user asfgit closed the pull request at: https://github.com/apache/flink/pull/4379
24. fixed in ea1edfb46f674035fd920c70100f60575600405f