Item 101
git_comments:

git_commits:

1. **summary:** Replace our AggregateValuesRule with Calcite's. (#3845)
   message: Replace our AggregateValuesRule with Calcite's. (#3845)

github_issues:

github_issues_comments:

github_pulls:

1. **title:** Replace our AggregateValuesRule with Calcite's.
   body: Our rule was contributed to Calcite in https://issues.apache.org/jira/browse/CALCITE-1489 and released in 1.11.0.
2. **title:** Replace our AggregateValuesRule with Calcite's.
   body: Our rule was contributed to Calcite in https://issues.apache.org/jira/browse/CALCITE-1489 and released in 1.11.0.
3. **title:** Replace our AggregateValuesRule with Calcite's.
   body: Our rule was contributed to Calcite in https://issues.apache.org/jira/browse/CALCITE-1489 and released in 1.11.0.

github_pulls_comments:

1. 👍

github_pulls_reviews:

jira_issues:

1. **summary:** Add rule, AggregateValuesRule, that applies to Aggregate on empty relation
   description: Add rule, AggegateValuesRule, that applies to Aggregate on empty relation. In CALCITE-1488, [~gian] wrote: {quote} I still have a problem with Aggregates that lack a group set, like SELECT COUNT\(*\) FROM s.foo WHERE 1 = 0. PruneEmptyRules doesn't reduce that, since the result is going to be one row, not empty. The ValuesReduceRules don't either, because they don't touch Aggregates. I ended up addressing this with a rule on my end: https://gist.github.com/gianm/1c192a47a7ce284be8af986f97e6dd8f Does that approach make sense? If so do you think it makes sense to contribute to Calcite? One thing I'm not sure about is if there's a better way to figure out what should be 0 and what should be NULL other than hard-coding COUNT and SUM0. {quote} Let's call it {{AggregateValuesRule}}, consistent with the naming convention where we start with the type of rel(s) being acted on. Initially it would apply to an empty {{Values}}, but potentially it could apply to one or more values. I don't recall aggregates having a way to tell us what they will return on empty (or constant) input. In future we could use the constant reducer for that, but for now, the rule should match a particular set of aggregates: {{COUNT}} and {{SUM0}} return zero; {{MIN}}, {{MAX}}, {{SUM}} return null. It must abort if it sees an aggregate it does not know how to handle.
2. **summary:** Add rule, AggregateValuesRule, that applies to Aggregate on empty relation
   description: Add rule, AggegateValuesRule, that applies to Aggregate on empty relation. In CALCITE-1488, [~gian] wrote: {quote} I still have a problem with Aggregates that lack a group set, like SELECT COUNT\(*\) FROM s.foo WHERE 1 = 0. PruneEmptyRules doesn't reduce that, since the result is going to be one row, not empty. The ValuesReduceRules don't either, because they don't touch Aggregates. I ended up addressing this with a rule on my end: https://gist.github.com/gianm/1c192a47a7ce284be8af986f97e6dd8f Does that approach make sense? If so do you think it makes sense to contribute to Calcite? One thing I'm not sure about is if there's a better way to figure out what should be 0 and what should be NULL other than hard-coding COUNT and SUM0. {quote} Let's call it {{AggregateValuesRule}}, consistent with the naming convention where we start with the type of rel(s) being acted on. Initially it would apply to an empty {{Values}}, but potentially it could apply to one or more values. I don't recall aggregates having a way to tell us what they will return

on empty (or constant) input. In future we could use the constant reducer for that, but for now, the rule should match a particular set of aggregates: {{COUNT}} and {{SUM0}} return zero; {{MIN}}, {{MAX}}, {{SUM}} return null. It must abort if it sees an aggregate it does not know how to handle.

3. **summary:** Add rule, AggregateValuesRule, that applies to Aggregate on empty relation
   description: Add rule, AggegateValuesRule, that applies to Aggregate on empty relation. In CALCITE-1488, [~gian] wrote: {quote} I still have a problem with Aggregates that lack a group set, like SELECT COUNT\(*\) FROM s.foo WHERE 1 = 0. PruneEmptyRules doesn't reduce that, since the result is going to be one row, not empty. The ValuesReduceRules don't either, because they don't touch Aggregates. I ended up addressing this with a rule on my end: https://gist.github.com/gianm/1c192a47a7ce284be8af986f97e6dd8f Does that approach make sense? If so do you think it makes sense to contribute to Calcite? One thing I'm not sure about is if there's a better way to figure out what should be 0 and what should be NULL other than hard-coding COUNT and SUM0. {quote} Let's call it {{AggregateValuesRule}}, consistent with the naming convention where we start with the type of rel(s) being acted on. Initially it would apply to an empty {{Values}}, but potentially it could apply to one or more values. I don't recall aggregates having a way to tell us what they will return on empty (or constant) input. In future we could use the constant reducer for that, but for now, the rule should match a particular set of aggregates: {{COUNT}} and {{SUM0}} return zero; {{MIN}}, {{MAX}}, {{SUM}} return null. It must abort if it sees an aggregate it does not know how to handle.
4. **summary:** Add rule, AggregateValuesRule, that applies to Aggregate on empty relation
   description: Add rule, AggegateValuesRule, that applies to Aggregate on empty relation. In CALCITE-1488, [~gian] wrote: {quote} I still have a problem with Aggregates that lack a group set, like SELECT COUNT\(*\) FROM s.foo WHERE 1 = 0. PruneEmptyRules doesn't reduce that, since the result is going to be one row, not empty. The ValuesReduceRules don't either, because they don't touch Aggregates. I ended up addressing this with a rule on my end: https://gist.github.com/gianm/1c192a47a7ce284be8af986f97e6dd8f Does that approach make sense? If so do you think it makes sense to contribute to Calcite? One thing I'm not sure about is if there's a better way to figure out what should be 0 and what should be NULL other than hard-coding COUNT and SUM0. {quote} Let's call it {{AggregateValuesRule}}, consistent with the naming convention where we start with the type of rel(s) being acted on. Initially it would apply to an empty {{Values}}, but potentially it could apply to one or more values. I don't recall aggregates having a way to tell us what they will return on empty (or constant) input. In future we could use the constant reducer for that, but for now, the rule should match a particular set of aggregates: {{COUNT}} and {{SUM0}} return zero; {{MIN}}, {{MAX}}, {{SUM}} return null. It must abort if it sees an aggregate it does not know how to handle.
5. **summary:** Add rule, AggregateValuesRule, that applies to Aggregate on empty relation
   description: Add rule, AggegateValuesRule, that applies to Aggregate on empty relation. In CALCITE-1488, [~gian] wrote: {quote} I still have a problem with Aggregates that lack a group set, like SELECT COUNT\(*\) FROM s.foo WHERE 1 = 0. PruneEmptyRules doesn't reduce that, since the result is going to be one row, not empty. The ValuesReduceRules don't either, because they don't touch Aggregates. I ended up addressing this with a rule on my end: https://gist.github.com/gianm/1c192a47a7ce284be8af986f97e6dd8f Does that approach make sense? If so do you think it makes sense to contribute to Calcite? One thing I'm not sure about is if there's a better way to figure out what should be 0 and what should be NULL other than hard-coding COUNT and SUM0. {quote} Let's call it {{AggregateValuesRule}}, consistent with the naming convention where we start with the type of rel(s) being acted on. Initially it would apply to an empty {{Values}}, but potentially it could apply to one or more values. I don't recall aggregates having a way to tell us what they will return on empty (or constant) input. In future we could use the constant reducer for that, but for now, the rule should match a particular set of aggregates: {{COUNT}} and {{SUM0}} return zero; {{MIN}}, {{MAX}}, {{SUM}} return null. It must abort if it sees an aggregate it does not know how to handle.
   label: code-design
6. **summary:** Add rule, AggregateValuesRule, that applies to Aggregate on empty relation
   description: Add rule, AggegateValuesRule, that applies to Aggregate on empty relation. In CALCITE-1488, [~gian] wrote: {quote} I still have a problem with Aggregates that lack a group set, like SELECT COUNT\(*\) FROM s.foo WHERE 1 = 0. PruneEmptyRules doesn't reduce that, since the result is going to be one row, not empty. The ValuesReduceRules don't either, because they don't touch Aggregates. I ended up addressing this with a rule on my end: https://gist.github.com/gianm/1c192a47a7ce284be8af986f97e6dd8f Does that approach make sense? If so do you think it makes sense to contribute to Calcite? One thing I'm not sure about is if there's a better way to figure out what should be 0 and what should be NULL other than hard-coding COUNT and SUM0. {quote} Let's call it {{AggregateValuesRule}}, consistent with the naming convention where we start

with the type of rel(s) being acted on. Initially it would apply to an empty {{Values}}, but potentially it could apply to one or more values. I don't recall aggregates having a way to tell us what they will return on empty (or constant) input. In future we could use the constant reducer for that, but for now, the rule should match a particular set of aggregates: {{COUNT}} and {{SUM0}} return zero; {{MIN}}, {{MAX}}, {{SUM}} return null. It must abort if it sees an aggregate it does not know how to handle.

7. **summary:** Add rule, AggregateValuesRule, that applies to Aggregate on empty relation
   description: Add rule, AggegateValuesRule, that applies to Aggregate on empty relation. In CALCITE-1488, [~gian] wrote: {quote} I still have a problem with Aggregates that lack a group set, like SELECT COUNT\(*\) FROM s.foo WHERE 1 = 0. PruneEmptyRules doesn't reduce that, since the result is going to be one row, not empty. The ValuesReduceRules don't either, because they don't touch Aggregates. I ended up addressing this with a rule on my end: https://gist.github.com/gianm/1c192a47a7ce284be8af986f97e6dd8f Does that approach make sense? If so do you think it makes sense to contribute to Calcite? One thing I'm not sure about is if there's a better way to figure out what should be 0 and what should be NULL other than hard-coding COUNT and SUM0. {quote} Let's call it {{AggregateValuesRule}}, consistent with the naming convention where we start with the type of rel(s) being acted on. Initially it would apply to an empty {{Values}}, but potentially it could apply to one or more values. I don't recall aggregates having a way to tell us what they will return on empty (or constant) input. In future we could use the constant reducer for that, but for now, the rule should match a particular set of aggregates: {{COUNT}} and {{SUM0}} return zero; {{MIN}}, {{MAX}}, {{SUM}} return null. It must abort if it sees an aggregate it does not know how to handle.
8. **summary:** Add rule, AggregateValuesRule, that applies to Aggregate on empty relation
   description: Add rule, AggegateValuesRule, that applies to Aggregate on empty relation. In CALCITE-1488, [~gian] wrote: {quote} I still have a problem with Aggregates that lack a group set, like SELECT COUNT\(*\) FROM s.foo WHERE 1 = 0. PruneEmptyRules doesn't reduce that, since the result is going to be one row, not empty. The ValuesReduceRules don't either, because they don't touch Aggregates. I ended up addressing this with a rule on my end: https://gist.github.com/gianm/1c192a47a7ce284be8af986f97e6dd8f Does that approach make sense? If so do you think it makes sense to contribute to Calcite? One thing I'm not sure about is if there's a better way to figure out what should be 0 and what should be NULL other than hard-coding COUNT and SUM0. {quote} Let's call it {{AggregateValuesRule}}, consistent with the naming convention where we start with the type of rel(s) being acted on. Initially it would apply to an empty {{Values}}, but potentially it could apply to one or more values. I don't recall aggregates having a way to tell us what they will return on empty (or constant) input. In future we could use the constant reducer for that, but for now, the rule should match a particular set of aggregates: {{COUNT}} and {{SUM0}} return zero; {{MIN}}, {{MAX}}, {{SUM}} return null. It must abort if it sees an aggregate it does not know how to handle.
9. **summary:** Add rule, AggregateValuesRule, that applies to Aggregate on empty relation
   description: Add rule, AggegateValuesRule, that applies to Aggregate on empty relation. In CALCITE-1488, [~gian] wrote: {quote} I still have a problem with Aggregates that lack a group set, like SELECT COUNT\(*\) FROM s.foo WHERE 1 = 0. PruneEmptyRules doesn't reduce that, since the result is going to be one row, not empty. The ValuesReduceRules don't either, because they don't touch Aggregates. I ended up addressing this with a rule on my end: https://gist.github.com/gianm/1c192a47a7ce284be8af986f97e6dd8f Does that approach make sense? If so do you think it makes sense to contribute to Calcite? One thing I'm not sure about is if there's a better way to figure out what should be 0 and what should be NULL other than hard-coding COUNT and SUM0. {quote} Let's call it {{AggregateValuesRule}}, consistent with the naming convention where we start with the type of rel(s) being acted on. Initially it would apply to an empty {{Values}}, but potentially it could apply to one or more values. I don't recall aggregates having a way to tell us what they will return on empty (or constant) input. In future we could use the constant reducer for that, but for now, the rule should match a particular set of aggregates: {{COUNT}} and {{SUM0}} return zero; {{MIN}}, {{MAX}}, {{SUM}} return null. It must abort if it sees an aggregate it does not know how to handle.
10. **summary:** Add rule, AggregateValuesRule, that applies to Aggregate on empty relation
    description: Add rule, AggegateValuesRule, that applies to Aggregate on empty relation. In CALCITE-1488, [~gian] wrote: {quote} I still have a problem with Aggregates that lack a group set, like SELECT COUNT\(*\) FROM s.foo WHERE 1 = 0. PruneEmptyRules doesn't reduce that, since the result is going to be one row, not empty. The ValuesReduceRules don't either, because they don't touch Aggregates. I ended up addressing this with a rule on my end: https://gist.github.com/gianm/1c192a47a7ce284be8af986f97e6dd8f Does that approach make sense? If so do you think it makes sense to contribute to Calcite? One thing I'm not sure about is if there's a better way to figure out what should be 0 and what should be NULL other than hard-coding COUNT and SUM0.

{quote} Let's call it {{AggregateValuesRule}}, consistent with the naming convention where we start with the type of rel(s) being acted on. Initially it would apply to an empty {{Values}}, but potentially it could apply to one or more values. I don't recall aggregates having a way to tell us what they will return on empty (or constant) input. In future we could use the constant reducer for that, but for now, the rule should match a particular set of aggregates: {{COUNT}} and {{SUM0}} return zero; {{MIN}}, {{MAX}}, {{SUM}} return null. It must abort if it sees an aggregate it does not know how to handle.
label: code-design

11. **summary:** Add rule, AggregateValuesRule, that applies to Aggregate on empty relation
description: Add rule, AggegateValuesRule, that applies to Aggregate on empty relation. In CALCITE-1488, [~gian] wrote: {quote} I still have a problem with Aggregates that lack a group set, like SELECT COUNT\(*\) FROM s.foo WHERE 1 = 0. PruneEmptyRules doesn't reduce that, since the result is going to be one row, not empty. The ValuesReduceRules don't either, because they don't touch Aggregates. I ended up addressing this with a rule on my end: https://gist.github.com/gianm/1c192a47a7ce284be8af986f97e6dd8f Does that approach make sense? If so do you think it makes sense to contribute to Calcite? One thing I'm not sure about is if there's a better way to figure out what should be 0 and what should be NULL other than hard-coding COUNT and SUM0. {quote} Let's call it {{AggregateValuesRule}}, consistent with the naming convention where we start with the type of rel(s) being acted on. Initially it would apply to an empty {{Values}}, but potentially it could apply to one or more values. I don't recall aggregates having a way to tell us what they will return on empty (or constant) input. In future we could use the constant reducer for that, but for now, the rule should match a particular set of aggregates: {{COUNT}} and {{SUM0}} return zero; {{MIN}}, {{MAX}}, {{SUM}} return null. It must abort if it sees an aggregate it does not know how to handle.

jira_issues_comments:

1. Hi Julian, Could you elaborate a little more on what this rule will do and why is this required ?
2. Sorry, I forgot to include the preceding discussion from CALCITE-1488. I just added it to the description. Hopefully it makes sense now.
3. Thanks [~julianhyde] it makes sense now
4. Patch in: https://github.com/apache/calcite/pull/324 I applied this to my local fork too, and it's working for me.
5. **body:** [~julianhyde] do you think this rule should replace PruneEmptyRules.AGGREGATE_INSTANCE? In that we get rid of that one, and make AggregateValuesRule handle both empty and nonempty groupSets.
label: code-design
6. I think we should keep both. * The grand-total case returns 1 row, which is the total of 0 rows. It performs something akin to constant reduction. Handled by the new {{AggregateValuesRule}}. * The non-grand-total case returns 0 rows and therefore truly "prunes" the tree. Handled by the existing {{PruneEmptyRules#AGGREGATE_INSTANCE}}.
7. Okay, cool. Then I think the patch in #324 should work.
8. Fixed in http://git-wip-us.apache.org/repos/asf/calcite/commit/3f92157d. Thanks for the PR, [~gian]!
9. Resolved in release 1.11.0 (2017-01-11).