

Spark SQL 作业实践

1.为Spark SQL添加一条自定义命令

- SHOW VERSION;
- 显示当前Spark版本和Java版本

在sqlbase.g4文件中增加新Keyword: SHOW VERSION, 并编译antlr4。

增加show version命令

```
| SET .*? #setConfiguration
| SET .*? #setConfiguration
| RESET configKey #resetQuotedConfiguration
| RESET .*? #resetConfiguration
| SHOW VERSION #showSparkAndJavaVersion
| unsupportedHiveNativeCommands .*? #failNativeCommand
;
```

增加version命令

```
1772 VALUES: 'VALUES';
1773 VIEW: 'VIEW';
1774 VIEWS: 'VIEWS';
1775 VERSION: 'VERSION';
1776 WHEN: 'WHEN';
1777 WHERE: 'WHERE';
1778 WINDOW: 'WINDOW';
1779 WITH: 'WITH';
1780 YEAR: 'YEAR';
1781 ZONE: 'ZONE';
```

在nonkeyword和ansinonkeyword中增加保留字

```
1504 | VALUES
1505 | VIEW
1506 | VIEWS
1507 | VERSION
1508 | WHEN
1509 | WHERE
1510 | WINDOW
```

编译antlr4

```
org.antlr4-maven-plugin:4.8:antlr4...
ms [INFO] --- antlr4-maven-plugin:4.8:antlr4 (default-cli) @ spark-catalyst_2.12 ---
[INFO] ANTLR 4: Processing source directory /Users/chenhao/github_code/spark/sql/catalyst/src/main/antlr4
[INFO] Processing grammar: org/apache/spark/sql/catalyst/parser/SqlBase.g4
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 22.596 s
[INFO] Finished at: 2021-09-05T22:33:54+08:00
[INFO] -----
```

在sparkSqlParse.scala文件中增加visitShowVersion方法

```
//noinspection ScalaStyle
override def visitShowVersion(ctx:
ShowSparkAndJavaVersionContext): LogicalPlan = withOrigin(ctx) {
    showVersionCommand()
}
```

新增文件sql/core/src/main/scala/org/apache/spark/sql/execution/command/showVersionCommand.scala, 实现showVersionCommand方法

```
package org.apache.spark.sql.execution.command

import org.apache.spark.sql.{Row, SparkSession}
import org.apache.spark.sql.catalyst.expressions.{Attribute,
AttributeReference}
import org.apache.spark.sql.types.StringType

case class showVersionCommand() extends LeafRunnableCommand {

    override val output: Seq[Attribute] =
        Seq(AttributeReference("user", StringType, nullable =
true)())

    //noinspection ScalaStyle
    override def run(sparkSession: SparkSession) : Seq[Row] ={
        val outputString = System.getenv("SPARK_VERSION")
        Seq(Row(outputString))
    }
}
```

待sbt编译成功后，测试show version命令是否可在spark-sql交互界面里成功运行。

首先直接运行成功，但会返回空值。

```
21/09/06 07:22:09 WARN ObjectStore: Version information not found in meta
21/09/06 07:22:09 WARN ObjectStore: setMetaStoreSchemaVersion called but
21/09/06 07:22:09 WARN ObjectStore: Failed to get database default, return
spark-sql>
>
> show version;
NULL
spark-sql> █
```

其次先设置好SPARK_VERSION环境变量，再在spark-sql交互界面运行，成功返回获取的版本号。

```
21/09/06 07:26:43 WARN ObjectStore: Version information not
21/09/06 07:26:43 WARN ObjectStore: setMetaStoreSchemaVersi
spark-sql>
> show version;
spark_3.1.2
spark-sql> █
```

2.构建SQL满足如下要求

通过set spark.sql.planChangeLog.level=WARN;查看

1. 构建一条SQL，同时apply下面三条优化规则：

- CombineFilters
- CollapseProject
- BooleanSimplification

-- 建表

```
drop table if exists test1;
```

```

create table test1(
    id int,
    name string,
    score int
);
-- 记录初始化
insert overwrite table test1
select 1 as id, '张三' as name, 60 as score
union all
select 2 as id, '李四' as name, 70 as score
union all
select 3 as id, '王五' as name, 80 as score;

```

执行以下SQL语句，观察优化规则。

```

select a.name,a.stu_score
from ( select name, score as stu_score from test1
      where 1='1') a
where a.stu_score=80.0;

```

apply结果如图

```

21/09/05 15:47:47 WARN PlanChangelogger:
-- Applying Rule org.apache.spark.sql.catalyst.optimizer.CollapseProject ==
Project [name#101, stu_score#99]                                     Project [name#101, score#102 AS stu_score#99]
+- Project [name#101, score#102 AS stu_score#99]                    +- Filter ((1 = cast(1 as int)) AND (score#102 = 80))
+- Filter ((1 = cast(1 as int)) AND (score#102 = 80))              +- HiveTableRelation ['default'.test1', org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#100, name#101, score#102], Partition Cols: []]
+- HiveTableRelation ['default'.test1', org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#100, name#101, score#102], Partition Cols: []]

```

```

21/09/05 15:49:16 WARN PlanChangelogger:
-- Applying Rule org.apache.spark.sql.catalyst.optimizer.BooleanSimplification ==
Project [name#107, score#108 AS stu_score#105]                     Project [name#107, score#108 AS stu_score#105]
+- Filter (true AND (cast(cast(score#108 as decimal(10,0)) as decimal(11,1)) = 80.0))
+- Filter (cast(cast(score#108 as decimal(10,0)) as decimal(11,1)) = 80.0)
+- HiveTableRelation ['default'.test1', org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#106, name#107, score#108], Partition Cols: []]
+- HiveTableRelation ['default'.test1', org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#106, name#107, score#108], Partition Cols: []]

```

2. 构建一条SQL，同时apply下面五条优化规则：

- ConstantFolding
- PushDownPredicates
- ReplaceDistinctWithAggregate
- ReplaceExceptWithAntiJoin

- FoldablePropagation

```
-- 建表
drop table if exists test2;
create table test2(
    id int,
    name string,
    sex string
);
drop table if exists test3;
create table test3(
    id int,
    course string,
    score int
);
-- 记录初始化
insert overwrite table test2
select 1 as id, '张三' as name, '男' as sex
union all
select 2 as id, '李四' as name, '男' as sex
union all
select 3 as id, '王五' as name, '男' as sex
union all
select 4 as id, '思思' as name, '女' as sex
union all
select 5 as id, '兰兰' as name, '女' as sex
union all
select 6 as id, null as name, null as sex;

insert overwrite table test3
select 1 as id, '语文' as course, 65 as score
union all
select 1 as id, '数学' as course, 70 as score
union all
select 1 as id, '英文' as course, 85 as score
union all
select 2 as id, '语文' as course, 65 as score
union all
select 2 as id, '数学' as course, 75 as score
union all
select 2 as id, '英文' as course, 90 as score
```

```

union all
select 3 as id, '语文' as course, 90 as score
union all
select 3 as id, '数学' as course, 80 as score
union all
select 3 as id, '英文' as course, 85 as score
union all
select 4 as id, '语文' as course, 80 as score
union all
select 4 as id, '数学' as course, 70 as score
union all
select 4 as id, '英文' as course, 85 as score;

```

执行以下SQL语句，观察优化规则。

```

select distinct t3.id
from (select id, sum(score) as allscore
      from test3
      group by id) as t3
join test2 as t2
  on t3.id = t2.id and t2.sex='男'
where t3.allscore >= 200

```

apply结果如图

```

21/09/05 21:54:58 WARN PlanChangelogger:
=== Applying Rule org.apache.spark.sql.catalyst.optimizer ConstantFolding ===
Aggregate [id#64], [id#64]
+- Project [id#64]
   +- Join Inner, ((id#64 = id#67) AND (sex#69 = 男))
      :- Project [id#64]
         :- Filter (allscore#63L >= cast(200 as bigint))
            :- +- Aggregate [id#64], [id#64, sum(cast(score#66 as bigint)) AS allscore#63L]
               :- +- Project [id#64, score#66]
                  :- +- HiveTableRelation [ 'default'.test3', org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#64, course#65, score#66], Partition Cols: [] ]
            +- Project [id#67, sex#69]
               +- HiveTableRelation [ 'default'.test2', org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#67, name#68, sex#69], Partition Cols: [] ]
      +- Project [id#64]
         :- Filter (allscore#63L >= 200)
            :- +- Aggregate [id#64], [id#64, sum(cast(score#66 as bigint)) AS allscore#63L]
               :- +- Project [id#64, score#66]
                  :- +- HiveTableRelation [ 'default'.test3', org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#64, course#65, score#66], Partition Cols: [] ]
            +- Project [id#67, sex#69]
               +- HiveTableRelation [ 'default'.test2', org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#67, name#68, sex#69], Partition Cols: [] ]

```

```

21/09/05 21:54:58 WARN PlanChangeLogger:
=== Applying Rule org.apache.spark.sql.catalyst.optimizer.PushDownPredicates ===
Aggregate [id#64], [id#64]
+- Project [id#64]
  +- Join Inner, (id#64 = id#67)
  :- Filter (isnotnull(id#64))
  :- Project [id#64]
  :- Filter (isnotnull(allscore#63L) AND (allscore#63L >= 200))
  :- Aggregate [id#64], [id#64, sum(cast(score#66 as bigint)) AS allscore#63L]
  :- Project [id#64, score#66]
course#65, score#66], Partition Cols: []
+- HiveTableRelation [ default, test3, org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#64, course#65, score#66], Partition Cols: []
+- Project [id#67]
+- Filter (isnotnull(id#67))
+- HiveTableRelation [ default, test2, org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#67, name#68, sex#69], Partition Cols: []
+- Project [id#67]
sex#69], Partition Cols: []
+- Filter (isnotnull(sex#69) AND (sex#69 = 男))
+- HiveTableRelation [ default, test2, org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#67, name#68, sex#69], Partition Cols: []

21/09/05 21:54:58 WARN PlanChangeLogger:
=== Applying Rule org.apache.spark.sql.catalyst.optimizer.PushDownPredicates ===
Aggregate [id#64], [id#64]
+- Project [id#64]
  +- Join Inner, (id#64 = id#67)
  :- Project [id#64]
  :- Filter (isnotnull(allscore#63L) AND (allscore#63L >= 200)) AND isnotnull(id#64))
  :- Aggregate [id#64], [id#64, sum(cast(score#66 as bigint)) AS allscore#63L]
  :- Project [id#64, score#66]
  :- HiveTableRelation [ default, test3, org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#64, course#65, score#66], Partition Cols: []
  :- Project [id#67]
  :- HiveTableRelation [ default, test2, org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#67, name#68, sex#69], Partition Cols: []
score#66], Partition Cols: []
+- Filter (isnotnull(sex#69) AND (sex#69 = 男)) AND isnotnull(id#67))
+- HiveTableRelation [ default, test2, org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#67, name#68, sex#69], Partition Cols: []
+- Project [id#67]
+- Filter (isnotnull(sex#69) AND (sex#69 = 男)) AND isnotnull(id#67))
+- HiveTableRelation [ default, test2, org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#67, name#68, sex#69], Partition Cols: []
Partition Cols: []

```

```

21/09/05 21:54:58 WARN PlanChangeLogger:
=== Applying Rule org.apache.spark.sql.catalyst.optimizer.ReplaceDistinctWithAggregate ===
IDistinct
+- Project [id#64]
  +- Filter (allscore#63L >= cast(200 as bigint))
  +- Join Inner, ((id#64 = id#67) AND (sex#69 = 男))
  :- Aggregate [id#64], [id#64, sum(cast(score#66 as bigint)) AS allscore#63L]
  :- HiveTableRelation [ default, test3, org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#64, course#65, score#66], Partition Cols: []
  :- HiveTableRelation [ default, test2, org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#67, name#68, sex#69], Partition Cols: []
  :- HiveTableRelation [ default, test2, org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#67, name#68, sex#69], Partition Cols: []

21/09/05 21:54:58 WARN PlanChangeLogger:
=== Result of Batch Replace Operators ===
IDistinct
+- Project [id#64]
  +- Filter (allscore#63L >= cast(200 as bigint))
  +- Join Inner, ((id#64 = id#67) AND (sex#69 = 男))
  :- Aggregate [id#64], [id#64, sum(cast(score#66 as bigint)) AS allscore#63L]
  :- HiveTableRelation [ default, test3, org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#64, course#65, score#66], Partition Cols: []
  :- HiveTableRelation [ default, test2, org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#67, name#68, sex#69], Partition Cols: []
  :- HiveTableRelation [ default, test2, org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#67, name#68, sex#69], Partition Cols: []

Aggregate [id#64], [id#64]
+- Project [id#64]
  +- Filter (allscore#63L >= cast(200 as bigint))
  +- Join Inner, ((id#64 = id#67) AND (sex#69 = 男))
  :- Aggregate [id#64], [id#64, sum(cast(score#66 as bigint)) AS allscore#63L]
  :- HiveTableRelation [ default, test3, org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#64, course#65, score#66], Partition Cols: []
  :- HiveTableRelation [ default, test2, org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#67, name#68, sex#69], Partition Cols: []
  :- HiveTableRelation [ default, test2, org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#67, name#68, sex#69], Partition Cols: []

```

3.实现自定义优化规则（静默规则）

待实现