Default type coercion rules of Spark

This document is to explain the type-coercion rules of Spark in simple words. For details, please refer to the code in TypeCoercion.scala

1 Promote strings

- For Binary Arithmetic, if one of the children is String, cast it as Double type
- For Binary Comparison
  - String and Decimal, cast both as Double type
  - Atomic and String, cast as Atomic type
- For function Abs/Sum/Average/StddevPop/StddevSamp/UnaryMinus/UnaryPositive/VariancePop/VarianceSamp/Skewness/Kurtosis, convert the String type as Double type.

2 Decimal Precision

- Binary Arithmetic of Decimal and Decimal

Operation	Result Precision	Result Scale
e1 + e2	max(s1, s2) + max(p1-s1, p2-s2) + 1	max(s1, s2)
e1 - e2	max(s1, s2) + max(p1-s1, p2-s2) + 1	max(s1, s2)
e1 * e2	p1 + p2 + 1	s1 + s2
e1 / e2	p1 - s1 + s2 + max(6, s1 + p2 + 1)	max(6, s1 + p2 + 1)
e1 % e2	min(p1-s1, p2-s2) + max(s1, s2)	max(s1, s2)
e1 union e2	max(s1, s2) + max(p1-s1, p2-s2)	max(s1, s2)

- Binary Comparison of Integral type and Decimal
  - int_col > decimal_literal => int_col > floor(decimal_literal)
  - int_col >= decimal_literal => int_col >= ceil(decimal_literal)
  - int_col < decimal_literal => int_col < ceil(decimal_literal)
  - int_col <= decimal_literal => int_col <= floor(decimal_literal)
  - decimal_literal > int_col => ceil(decimal_literal) > int_col
  - decimal_literal >= int_col => floor(decimal_literal) >= int_col
  - decimal_literal < int_col => floor(decimal_literal) < int_col
  - decimal_literal <= int_col => ceil(decimal_literal) <= int_col

3 Boolean Equality

When comparing Boolean column and Numeric type column with EqualTo/EqualNullSafe,

- cast the boolean as the target numeric type.

4 Widen data type for the children of operators/functions

4.1 Applicable operators/functions

For the following operator/functions, apply the procedure in 4.2 or 4.3(if there are more than 2 children) to get the inferred result type. After that, add a Cast operator to any children which have different data type with the result one.

- In
- Except
- Intersect
- Union

- CreateArray
- Concat
- Sequence
- MapConcat
- CreateMap
- Greatest(exclude the step “Promote String”)
- Least(exclude the step “Promote String”)
- CaseWhen
- If

4.2 Procedure of finding the wider type of two input types

- 4.2.1 Find the tightest common type
  - Decimal and Integral, return the Integral type if it is wider
  - Numeric and Numeric, return the wider one as per the precedence list
  - Timestamp and Date, return the timestamp type
- 4.2.2 Find Wider Type For Decimal
  - Decimal and Decimal, return the wider one
  - Decimal and Integral, convert the Integral type to Decimal type and return the wider Decimal one
  - Decimal and Fractional(float/double), simply return double
- 4.2.3 Promote String
  - String and Atomic type, if the atomic type is not binary or boolean, return String type
- 4.2.4 For Array/Map/Struct type, apply the above steps recursively and get the result type.

4.3 Procedure of finding the widest type of multiple(> 2) input types

- If there is String type or nested String type in Array type, move it to the front of the input
- From left to right(foldLeft), perform the procedure in 4.1 to get the result type

5 Division

If the data type of Divide’s left child is neither Double nor Decimal, cast both left and right as Double type, e.g.

```
spark.sql("select 1/2").show()
+----------------------------------------+
|(CAST(1 AS DOUBLE) / CAST(2 AS DOUBLE))|
+----------------------------------------+
```

```
|                                     0.5       |
+-------------------------------------+
```

6 IntegralDivision

Cast both children to Long type.
e.g,

```
explain select 1 div 2;
== Physical Plan ==
*(1) Project [0 AS (CAST(1 AS BIGINT) div CAST(2 AS BIGINT))#13L]
+- *(1) Scan OneRowRelation[]
```

7 Implicit cast

- For binary operators, if neither side is decimal type, perform the procedure “4.2.1 Find the tightest common type” to get the result type. After that, add cast in children if needed.
- For expressions with trait ImplicitCastInputTypes(176 expressions in total), the expected input data types are specified in Spark. If the data types of the input children are different from the expected data types, the following implicit castings are performed:
    - (StringType, NumericType) => DoubleType
    - (d: NumericType, DecimalType) => DecimalType.forType(d)
    - (_: NumericType, target: NumericType) => target
    - (DateType, TimestampType) => TimestampType
    - (TimestampType, DateType) => DateType
    - (StringType, DecimalType) => DecimalType(38, 18)
    - (StringType, DateType) => DateType
    - (StringType, TimestampType) => TimestampType
    - (StringType, BinaryType) => BinaryType
    - (any: AtomicType, StringType) => StringType

8 Datetime operations

- If the first child of DateAdd/DateSub is of String/Timestamp type, cast it as Date type
- If the first child of SubtractTimestamps is of String/Date type, cast it as Timestamp type
- If the first child of TimeAdd is of String type, cast it as Timestamp type.

9 Window frame coercion

Cast WindowFrame boundaries (BETWEEN lower AND upper) to the type they operate upon. (avoid return null)