Introduction to Hadoop Hive Data Types

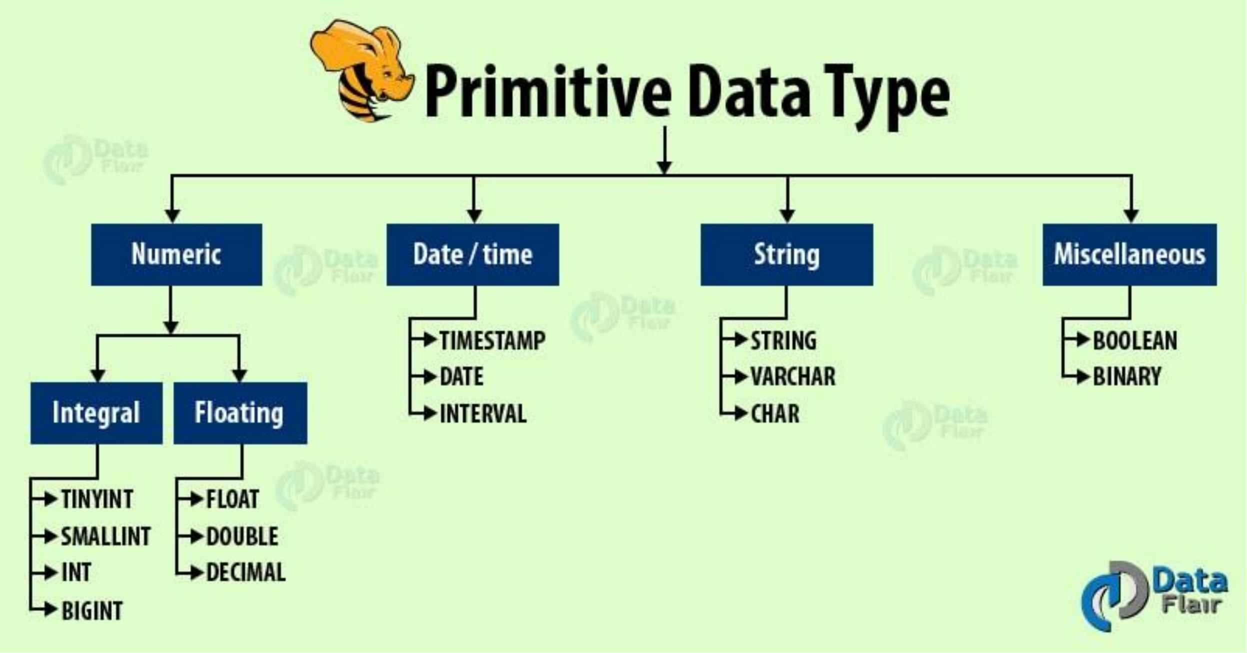
Data types are used for specifying the column/field type in Hive tables. Hive data types can be classified into following categories:

2.1. Primitive Data type

Primitive Data Types also divide into 4 types which are as follows:

- Numeric Data Type
- Date/Time Data Type
- String Data Type
- Miscellaneous Data Type

Let us now discuss these Hive Primitive data types one by one-



2.1.1. Numeric Data Type

The Hive Numeric Data types also classified into two types-

- Integral Data Types
- Floating Data Types

a) Integral Data Types

The Hive Integral data types are as follows-

- **TINYINT** (1-byte (8 bit) signed integer, from -128 to 127)
- **SMALLINT** (2-byte (16 bit) signed integer, from -32, 768 to 32, 767)
- **INT** (4-byte (32-bit) signed integer, from –2,147,483,648to 2,147,483,647)
- **BIGINT** (8-byte (64-bit) signed integer, from –9,223,372,036,854,775,808 to 9,223,372,036,854,775,807)

b) Floating Data Types

The Hive Floating data types are as follows-

- **FLOAT** (4-byte (32-bit) single-precision floating-point number)
- **DOUBLE** (8-byte (64-bit) double-precision floating-point number)
- **DECIMAL** (Arbitrary-precision signed decimal number)

2.1.2. Date/Time Data Type

The second category of Apache Hive primitive data type is Date/Time data types. The following data types comes into this category-

- **TIMESTAMP** (Timestamp with nanosecond precision)
- **DATE** (date)
- **INTERVAL**

2.1.3. String Data Type

String data types are the third category under Hive data types. Below are the data types that come into this-

- **STRING** (Unbounded variable-length character string)

- **VARCHAR** (Variable-length character string)
- **CHAR** (Fixed-length character string)

2.1.4. Miscellaneous Data Type

The two data types come into Hive miscellaneous data types-

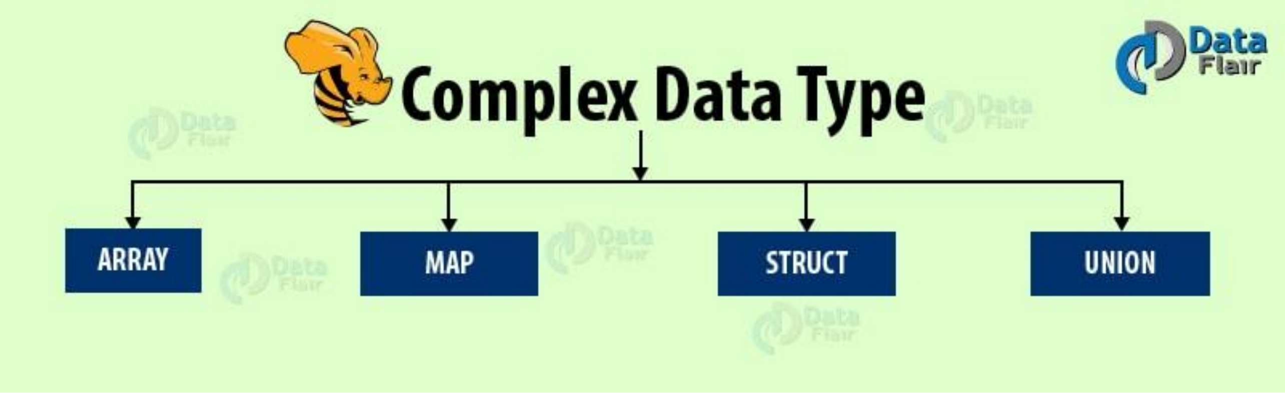
- **BOOLEAN** (True/false value)
- **BINARY** (Byte array)

2.2. Complex Data Type

In this category of Hive data types following data types are come-

- Array
- MAP
- STRUCT
- UNION

let's now discuss the Hive Complex data types with the example-



2.2.1. ARRAY

An ordered collection of fields. The fields must all be of the same type.

Syntax: ARRAY<data_type>

E.g. array (1, 2)

2.2.2. MAP

An unordered collection of **key-value pairs**. Keys must be primitives; values may be any type. For a particular map, the keys must be the same type, and the values must be the same type.

Syntax: MAP<primitive_type, data_type>

E.g. map('a', 1, 'b', 2).

2.2.3. STRUCT

A collection of named fields. The fields may be of different types.

Syntax: STRUCT<col_name : data_type [COMMENT col_comment],…..>

E.g. struct('a', 1 1.0),[b] named_struct('col1', 'a', 'col2', 1, 'col3', 1.0)

2.2.4. UNION

A value that may be one of a number of defined data   The value is tagged with an integer (zero-indexed) representing its data type in the union.

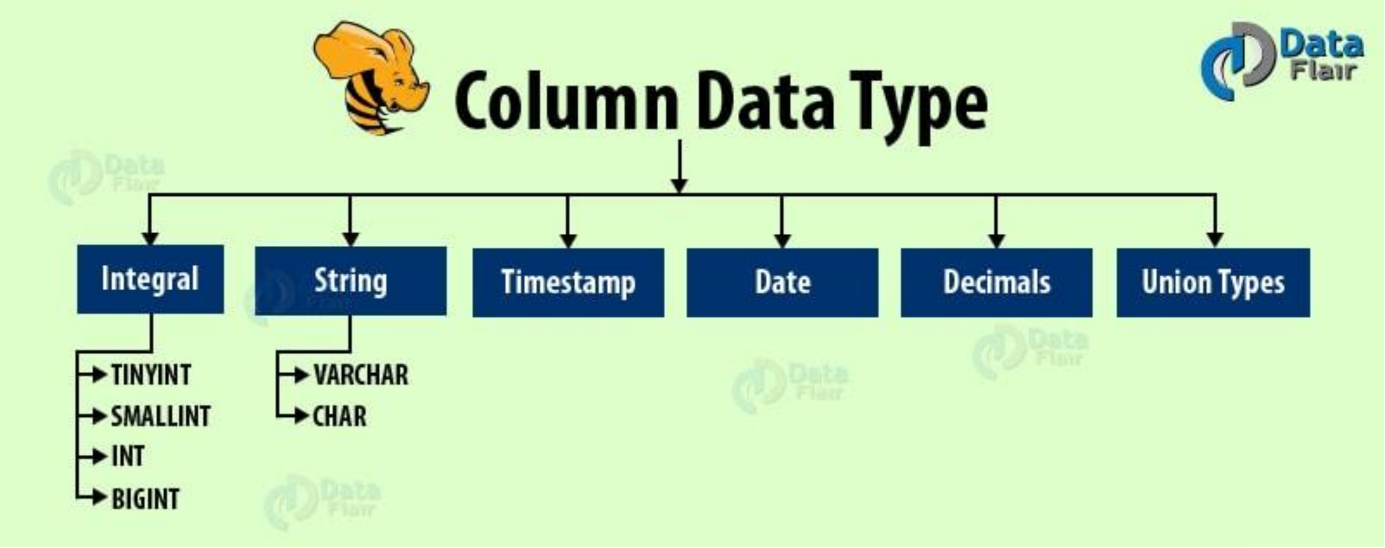
Syntax: UNIONTYPE<data_type, data_type, …>

E.g. create_union(1, 'a', 63)

2.3. Column Type

- Integral Type
- Strings
- Timestamp
- Dates
- Decimals
- Union Types

Let us discuss these Hive Column data types one by one-



2.3.1. Integral type

In this category of Hive data types following 4 data types are come-

- TINYINT
- SMALLINT
- INT/INTEGER
- BIGINT

By default, Integral literals are assumed to be **INT**. When the data range exceeds the range of INT, we need to use **BIGINT**. If the data range is smaller than the INT, we uses **SMALLINT**. And **TINYINT** is smaller than SMALLINT.

Table	Postfix	Example
TINYINT	Y	100Y
SMALLINT	S	100S

BIGINT	L	100L

2.3.2. Strings

The string can be specified with either single quotes (‘) or double quotes (“). Apache Hive use C-style escaping within the strings.

Data Type	Length
VARCHAR	1 to 65355
CHAR	255

a) VARCHAR

These are created with a length specifier (between 1 and 65355). It defines the maximum number of characters allowed in the character string.

b) Char

These are similar to VARCHAR. But they are fixed-length meaning that values shorter than the specified length value are padded with spaces but trailing spaces are not important during comparisons. 255 is the maximum fixed length.

2.3.3. Timestamp

Hive supports traditional UNIX timestamp with operational nanosecond precision. Timestamps in text files use format ”YYYY-MM-DD HH:MM:SS.fffffffff” and “yyyy-mm-dd hh:mm:ss.ffffffffff”.

2.3.4. Dates

DATE values are described in particular year/month/day (YYYY-MM-DD) format.E.g. DATE ‘2017-01-01’. These types don’t have a time of day component. This type supports range of values for 0000-01-01 to 9999--12-31.

2.3.5. Decimals

In Hive DECIMAL type is similar to Big Decimal format of java. This represents immutable arbitrary precision. The syntax and example are below:

Apache Hive 0.11 and 0.12 has the precision of the DECIMAL type fixed. And it's limited to 38 digits.

Apache Hive 0.13 users can specify scale and precision when creating tables with the DECIMAL data type using DECIMAL (precision, scale) syntax. If the scale is not specified, then it defaults to 0 (no fractional digits). If no precision is specified, then it defaults to 10.

```
CREATE TABLE foo (
a DECIMAL, -- Defaults to decimal(10,0)b DECIMAL(9, 7)
b DECIMAL(9, 7)
)
```

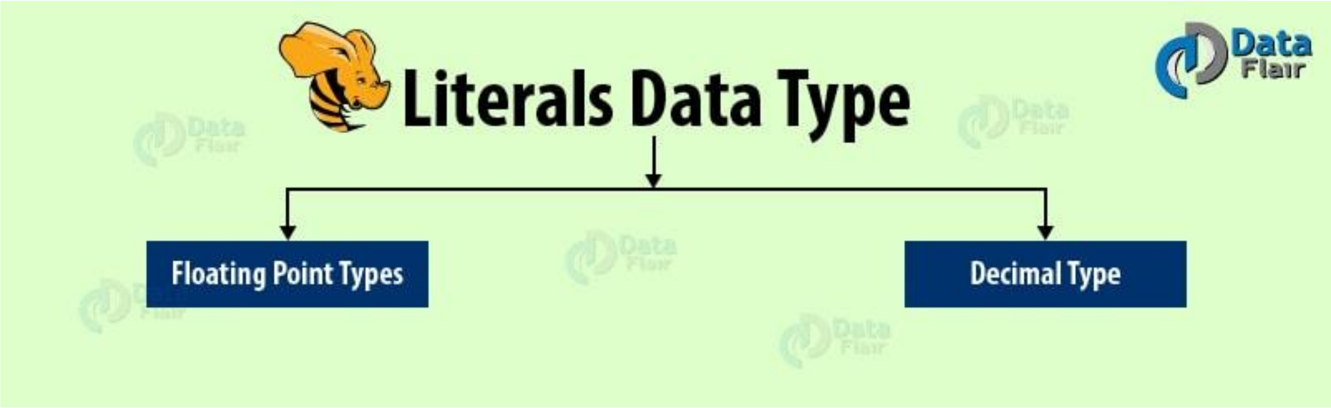
2.3.6. Union Types

It is the collection of heterogeneous data types. By using **create union** we can create an instance. The syntax and example are below:

```
CREATE TABLE union_test(foo UNIONTYPE<int, double, array<string>, struct<a:int,b:string>>);
SELECT foo FROM union_test;
{0:1}
{1:2.0}
{2:["three","four"]}
{3:{"a":5,"b":"five"}}
{2:["six","seven"]}
{3:{"a":8,"b":"eight"}}
{0:9}
{1:10.0}
```

2.4. Literals

In Hive following literals are used:



- Floating Point Types
- Decimal Type

2.4.1. Floating Point Types

These are nothing but numbers with decimal points. This type of data is composed of the DOUBLE data type.

2.4.2. Decimal Type

This type is nothing but floating point value with the higher range than the DOUBLE data type. The decimal type range is approximate -10^{-308} to 10^{308}.

2.5. Null Value

In Hive, missing values are represented by the special value **NULL**.