# 项目三 Hive基本语法



《Hive数据仓库应用教程》

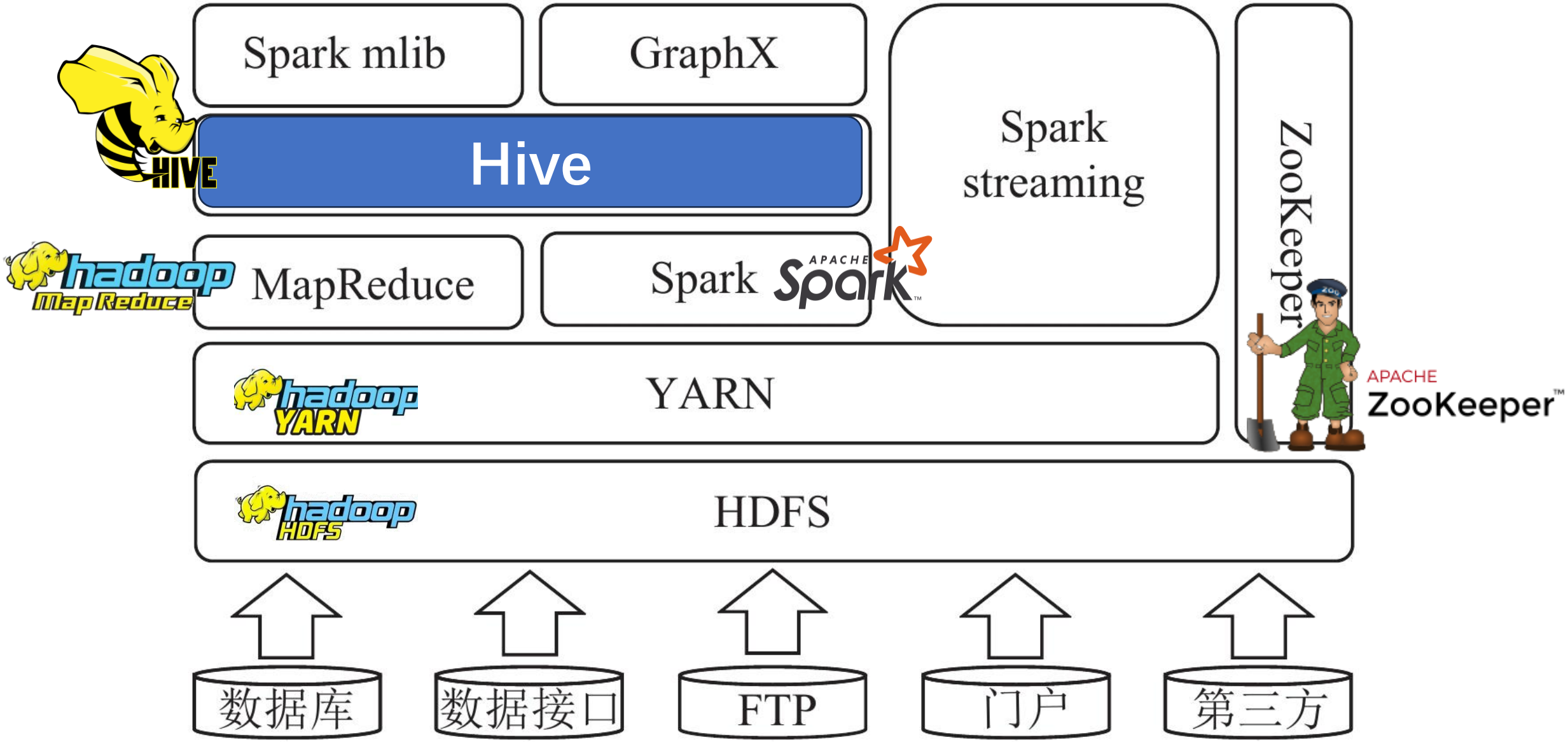
# 目 录 / CONTENTS


1 任务3-1 认识数据类型

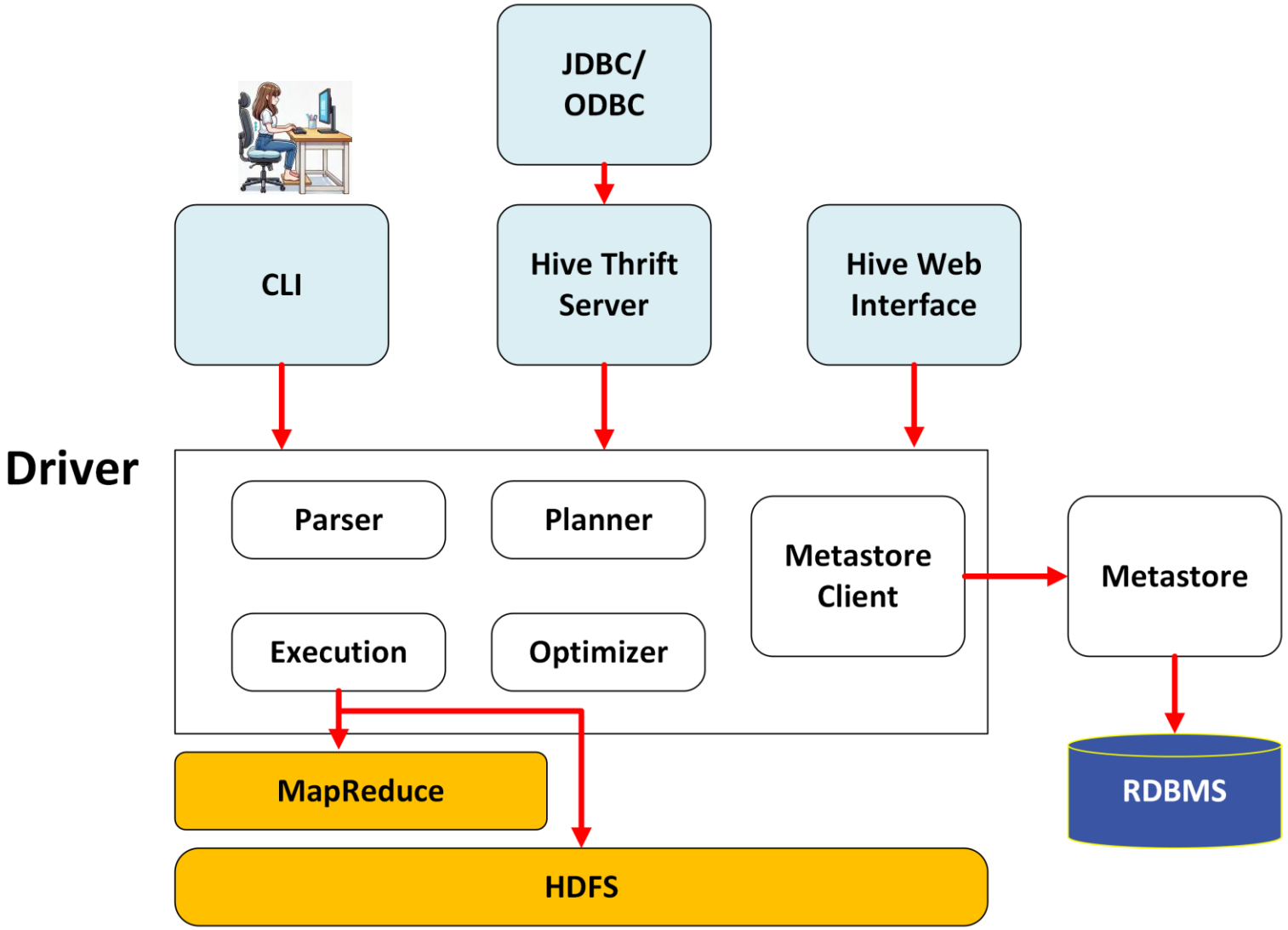
2 任务3-2 认识运算符

3 任务3-3 认识Hive命令

4 任务3-4 存储数据


Spark mlib
GraphX
Hive
Spark streaming
ZooKeeper
MapReduce
Spark
YARN
HDFS
数据库
数据接口
FTP
门户
第三方
APACHE ZooKeeper™

# Apache Hive Architecture



# 任务 3-1 认识数据类型

# 3.1.1 任务描述

在当前的任务中，我们将深入探讨
Hive原生数据类型、复杂数据类型及数据类型之间的转换。
数据表的设计与实现将直接影响数据库的运行效率。

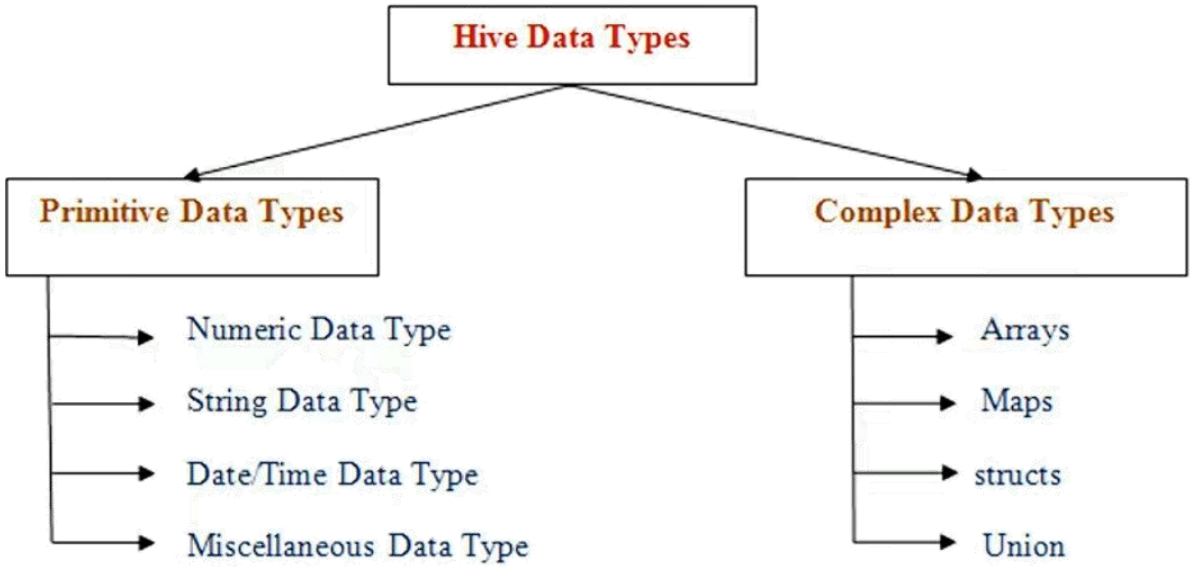
通过本次任务，
学生们将更好地理解Hive数据类型为认识和理解数据表结构打下基础。

# 3.1.2 Hive数据类型详解

## 整体概述

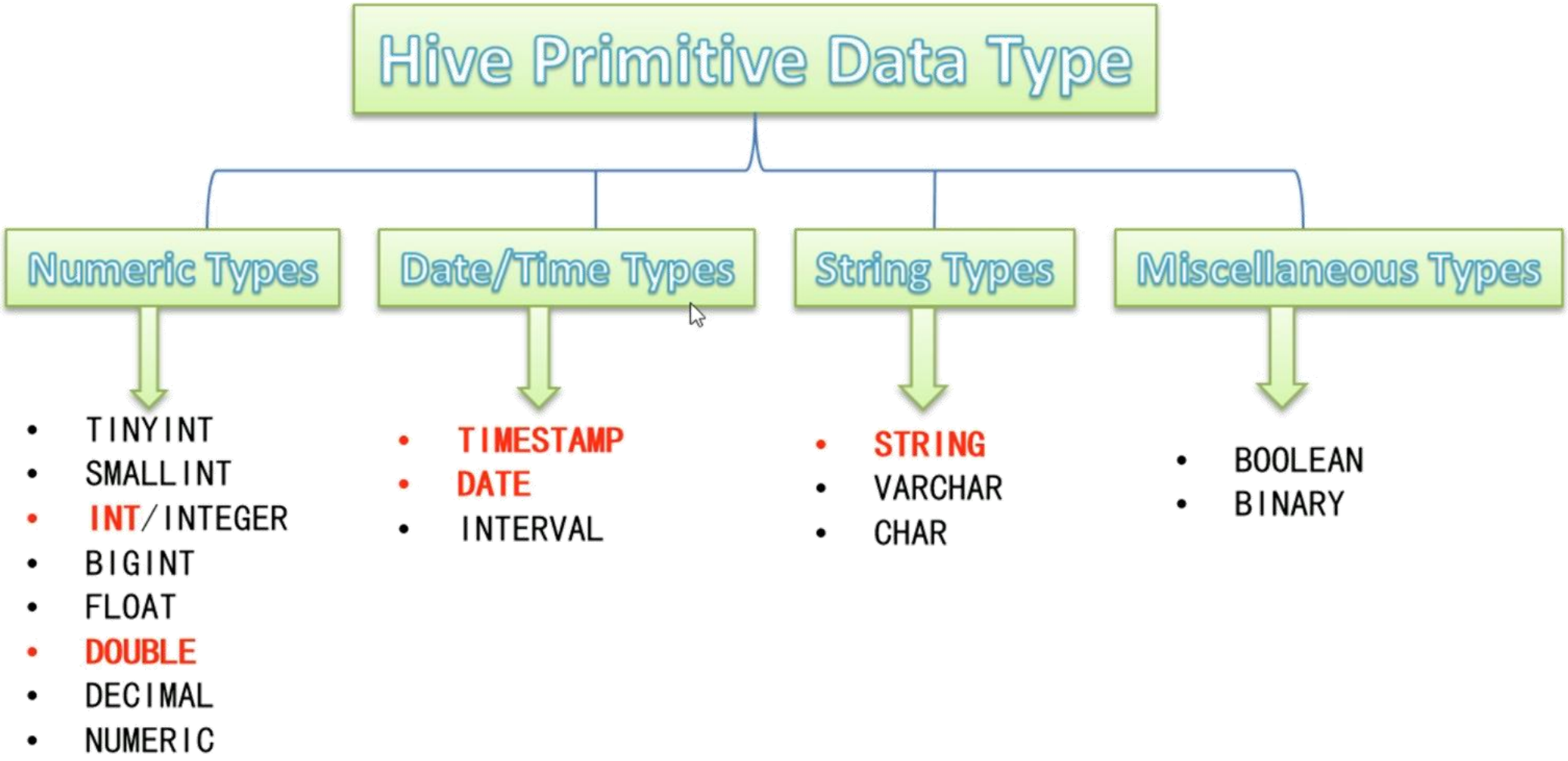
- Hive数据类型指的是表中列的字段类型;
- 整体分为两类：原生数据类型（primitive data type）和复杂数据类型（complex data type）。

- 原生数据类型包括：数值类型、时间日期类型、字符串类型、杂项数据类型;
- 复杂数据类型包括：array:数组、map:映射、struct:结构、union:联合体。



# 3.1.2 Hive数据类型详解

## 原生数据类型

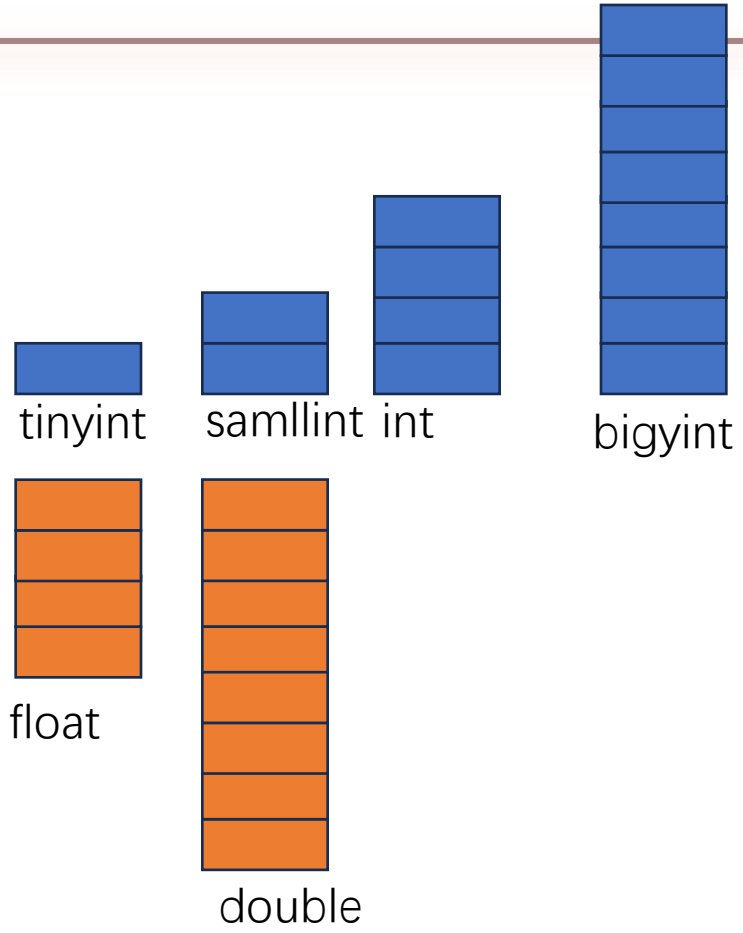


# 3.1.2 Hive数据类型详解

## 原生数据类型

### 数值类型（Numeric）

- TINYINT - 1字节整数（-128 ~ 127）
- SMALLINT - 2字节整数（-32,768 ~ 32,767）
- INT - 4字节整数（-2,147,483,648 ~ 2,147,483,647）
- BIGINT - 8字节整数
- FLOAT - 4字节单精度浮点数
- DOUBLE - 8字节双精度浮点数
- DECIMAL - 定点数（Hive 0.13.0之后版本）




### 字符串类型（String）

- STRING - 可变长度字符串
- VARCHAR - 带最大长度的可变长度字符串（Hive 0.12.0之后）
- CHAR - 固定长度字符串（Hive 0.13.0之后）

# 3.1.2 Hive数据类型详解

## 原生数据类型

### 日期/时间类型（Date/Time）

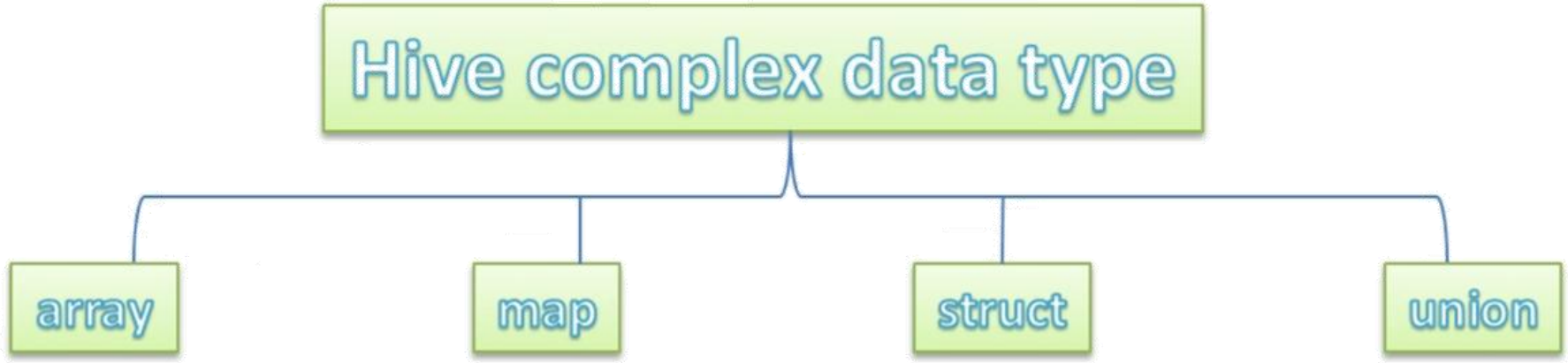
- **TIMESTAMP** - 日期和时间（Unix时间戳格式）:'2026-03-05 10:30:00'
- **DATE** - 日期（Hive 0.12.0之后）:'2026-03-05'

### 其他类型

- **BOOLEAN** - 布尔值（true/false）
- **BINARY** - 字节数组（Hive 0.8.0之后）

# 3.1.2 Hive数据类型详解

## 复杂数据类型




array<datatype>

struct<col_name : data_type ...>

map<key_type, value_type>

UNIONTYPE<data_type, data_type, ...>
Hive支持的还不是很完整

## 复杂数据类型

### ARRAY（数组）

- 相同类型元素的集合
  ARRAY<data_type>
  -- 示例：ARRAY<INT>, ARRAY<STRING>

### MAP（映射）

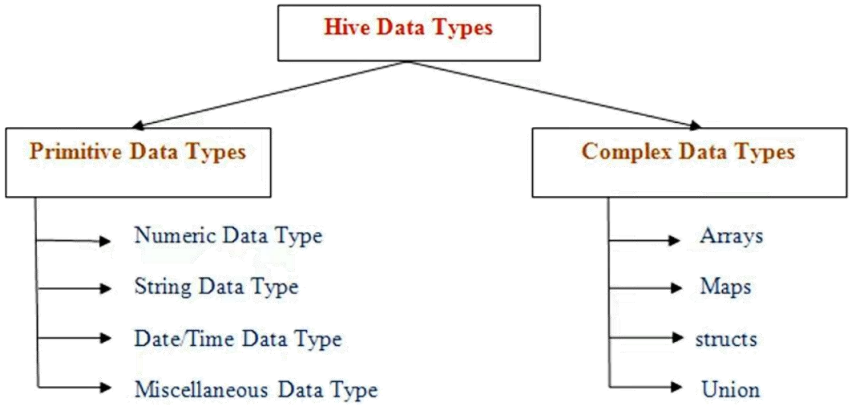
- 键值对集合（键必须是基本类型，值可以是任何类型）
  MAP<primitive_type, data_type>
  -- 示例： MAP<STRING, INT>

### STRUCT（结构体）

- 包含多个不同类型的字段的记录
  STRUCT<col_name : data_type, ...>
  *-- 示例：STRUCT<name:STRING, age:INT, salary:DOUBLE>*

## 注意事项

- 在Hive SQL中，数据类型的英文字母大小写不敏感；
- 除SQL数据类型外，还支持Java数据类型，比如字符串string；
- 复杂数据类型的使用通常需要与分隔符指定语法配合使用；
- 如果定义的数据类型与文件内容不一致，Hive会尝试隐式转换，但不保证成功。



## 3．使用示例

创建表示例

```
CREATE TABLE employees (
    id INT,
    name STRING,
    salary DOUBLE,
    is_active BOOLEAN,
    skills ARRAY<STRING>,                                    -- 技能数组
    dependents MAP<STRING, INT>,                             -- 家属映射（姓名->年龄）
    address STRUCT<street:STRING, city:STRING, zip:INT>, -- 地址结构体
    hire_date DATE
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','                                     -- 字段分隔符
COLLECTION ITEMS TERMINATED BY '|'                           -- 集合元素分隔符
MAP KEYS TERMINATED BY '#'                                   -- Map键值分隔符
STORED AS TEXTFILE;
```

3．使用示例(实践)

》hive-bee

>>show tables;
>>desc employees

```
0: jdbc:hive2://localhost:10000> create table employees(
0: jdbc:hive2://localhost:10000> id int,
0: jdbc:hive2://localhost:10000> name string,
0: jdbc:hive2://localhost:10000> salary double,
0: jdbc:hive2://localhost:10000> is_active boolean,
0: jdbc:hive2://localhost:10000> skills array<string>,
0: jdbc:hive2://localhost:10000> dependents map<string, int>,
0: jdbc:hive2://localhost:10000> address struct<street:string,  city:string, zip:int>,
0: jdbc:hive2://localhost:10000> hire_date date
0: jdbc:hive2://localhost:10000> )
0: jdbc:hive2://localhost:10000> ;
0: jdbc:hive2://localhost:10000> show tables;
+------------+
|  tab_name  |
+------------+
| dual       |
| employees  |
+------------+
0: jdbc:hive2://localhost:10000> describe employees;
+-------------+-----------------------------------------------+----------+
|  col_name   |                   data_type                   | comment  |
+-------------+-----------------------------------------------+----------+
| id          | int                                           |          |
| name        | string                                        |          |
| salary      | double                                        |          |
| is_active   | boolean                                       |          |
| skills      | array<string>                                 |          |
| dependents  | map<string,int>                               |          |
| address     | struct<street:string,city:string,zip:int>     |          |
| hire_date   | date                                          |          |
+-------------+-----------------------------------------------+----------+
```

## 3．使用示例(实践)

```
INSERT INTO employees VALUES
(
    1,
    '张伟',
    85000.50,
    true,
    ARRAY('Java', 'Python', 'SQL'),
    MAP('配偶', 35, '儿子', 8, '女儿', 5),
    NAMED_STRUCT('street', '中关村大街1号', 'city', '北京', 'zip', 100080),
    '2020-03-15'
),
(
    2,
    '李娜',
    92000.75,
    true,
    ARRAY('Hadoop', 'Spark', 'Hive', 'Kafka'),
    MAP('配偶', 38, '母亲', 65),
    NAMED_STRUCT('street', '浦东新区陆家嘴环路1000号', 'city', '上海', 'zip', 200120),
    '2019-07-22'
);
```

# 3.1.3 任务实施

## 3．使用示例(实践)

```
0: jdbc:hive2://localhost:10000> select * from employees;
+---------------+-----------------+-------------------+---------------------+-----------------------------------+-------------------------+-------------------------------------------------------+---------------------+
| employees.id  | employees.name  | employees.salary  | employees.is_active |        employees.skills          |  employees.dependents   |          employees.address                          | employees.hire_date  |
+---------------+-----------------+-------------------+---------------------+-----------------------------------+-------------------------+-------------------------------------------------------+---------------------+
| 1             | 张伟            | 85000.5           | true                | ["Java","Python","SQL"]           | {"配偶":35,"儿子":8,"女儿":5} | {"street":"中关村大街1号","city":"北京","zip":100080}          | 2020-03-15          |
| 2             | 李娜            | 92000.75          | true                | ["Hadoop","Spark","Hive","Kafka"] | {"配偶":38,"母亲":65}         | {"street":"浦东新区陆家嘴环路1000号","city":"上海","zip":200120} | 2019-07-22          |
+---------------+-----------------+-------------------+---------------------+-----------------------------------+-------------------------+-------------------------------------------------------+---------------------+
0: jdbc:hive2://localhost:10000>
```

```
0: jdbc:hive2://localhost:10000> select id, name, salary, is_active from employees;
+-----+-------+-----------+------------+
| id  | name  |  salary   | is_active  |
+-----+-------+-----------+------------+
| 1   | 张伟    | 85000.5   | true       |
| 2   | 李娜    | 92000.75  | true       |
+-----+-------+-----------+------------+
0: jdbc:hive2://localhost:10000>
```

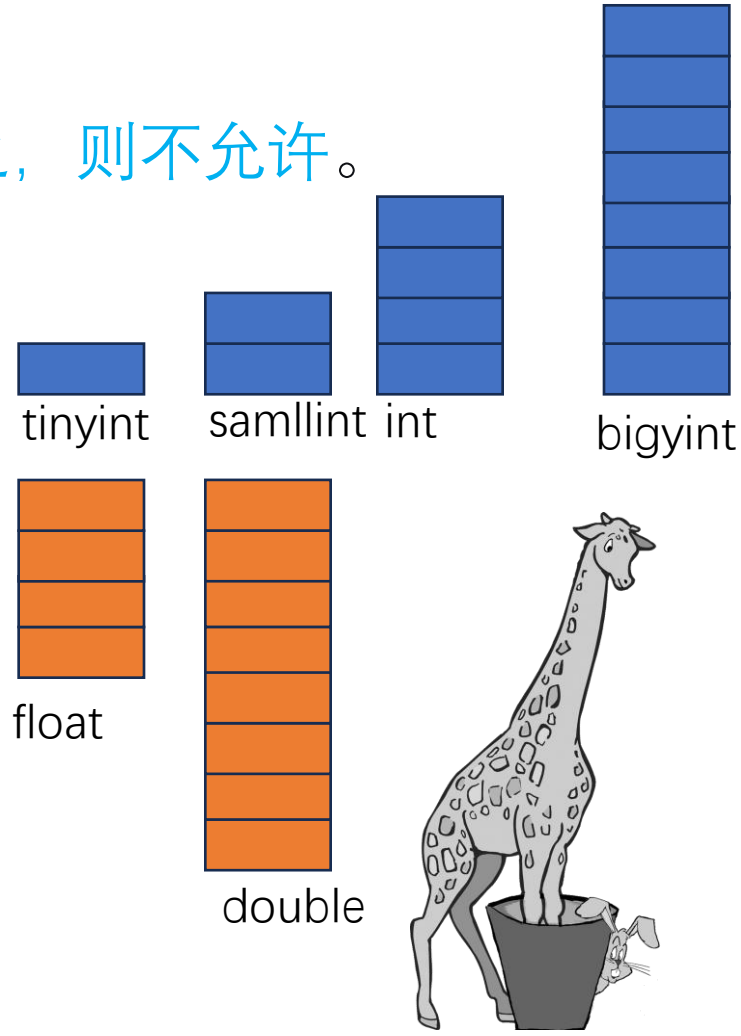
# 3.1.4 Hive类型转换

## 隐式转换(implicit transformation)

- 与标准SQL类似，HQL支持隐式和显式类型转换。
- 原生类型从窄类型到宽类型的转换称为隐式转换，反之，则不允许。
- 下表描述了类型之间允许的隐式转换：

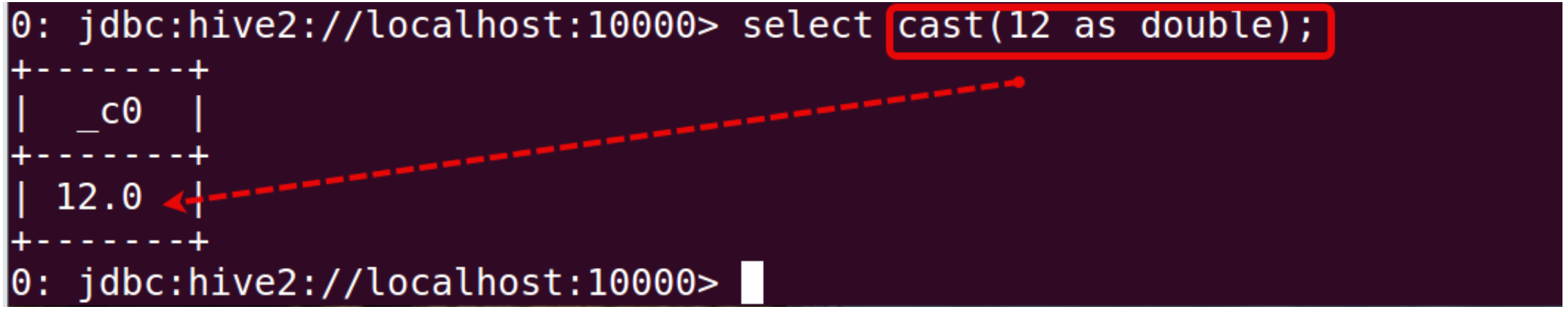
Allowed Implicit Conversions

| | void | boolean | tinyint | smallint | int | bigint | float | double |
|---|---|---|---|---|---|---|---|---|
| **void to** | true | true | true | true | true | true | true | true |
| **boolean to** | false | true | false | false | false | false | false | false |
| **tinyint to** | false | false | true | true | true | true | true | true |
| **smallint to** | false | false | false | true | true | true | true | true |
| **int to** | false | false | false | false | true | true | true | true |
| **bigint to** | false | false | false | false | false | true | true | true |
| **float to** | false | false | false | false | false | false | true | true |



## 显式转换（explicit transformation）

- 显式类型转换使用CAST函数。
  - 例如，CAST('100'as INT)会将100字符串转换为100整数值。
- 如果强制转换失败，例如CAST（'Allen' as INT），该函数返回NULL。



## 5．Hive类型转换（实践）

```
0: jdbc:hive2://localhost:10000> select 1+2.5;
+------+
| _c0  |
+------+
| 3.5  |
+------+
0: jdbc:hive2://localhost:10000> select cast('123' as int);
+------+
| _c0  |
+------+
| 123  |
+------+
0: jdbc:hive2://localhost:10000> select cast(salary as string) from employees;
+-----------+
|  salary   |
+-----------+
| 85000.5   |
| 92000.75  |
+-----------+
0: jdbc:hive2://localhost:10000> select skills[0] from employees;
+---------+
|   _c0   |
+---------+
| Java    |
| Hadoop  |
+---------+
0: jdbc:hive2://localhost:10000> select dependents['儿子'] from employees;
+-------+
|  _c0  |
+-------+
| 8     |
| NULL  |
+-------+
0: jdbc:hive2://localhost:10000> select address.city from employees;
+-------+
| city  |
+-------+
| 北京    |
| 上海    |
+-------+
```

# 3.1.6 Hive数据类型测验

**Hive数据类型测验（4选1）**

**问题1**

以下哪个是Hive中的有效数值类型？

A) NUMBER

B) INTEGER

C) NUMERIC

D) TINYINT

**答案：**

# 3.1.6 Hive数据类型测验

**Hive数据类型测验（4选1）**

**问题2**

在Hive中，可以存储键值对集合的数据类型是什么？

A) ARRAY

B) SET

C) MAP

D) DICTIONARY

**答案：**

# 3.1.6 Hive数据类型测验

**Hive数据类型测验（4选1）**

**问题3**

关于Hive的STRING和VARCHAR数据类型，以下哪个说法是正确的？

A) STRING和VARCHAR没有区别，可以互换使用
B) VARCHAR有最大长度限制，STRING没有
C) STRING有最大长度限制，VARCHAR没有
D) 两者都不支持中文字符存储

**答案：**

# 3.1.6 Hive数据类型测验

**Hive数据类型测验（4选1）**

**问题4**

以下哪个Hive数据类型可以存储不同类型的字段组合？

A) ARRAY<STRUCT>

B) MAP

C) UNIONTYPE

D) STRUCT

**答案：**

# 3.1.6 Hive数据类型测验

**Hive数据类型测验（4选1）**

**问题5**

创建包含数组类型的表时，哪个子句用于指定数组元素之间的分隔符？

A) FIELDS TERMINATED BY
B) COLLECTION ITEMS TERMINATED BY
C) MAP KEYS TERMINATED BY
D) ARRAY ELEMENTS TERMINATED BY

**答案：**

# 3.1.6 Hive数据类型测验

## Hive数据类型测验（4选1）

### 问题6 (新增：显式转换问题)

在Hive中，将字符串'123.45'转换为DECIMAL(5,2)类型的正确语法是什么？

A) TO_DECIMAL('123.45', 5, 2)
B) DECIMAL('123.45', 5, 2)
C) CAST('123.45' AS DECIMAL(5,2))
D) CONVERT('123.45', DECIMAL(5,2))

# 感谢您的观看



《Hive数据仓库应用教程》