BigData Concepts

04 January 2024 09:25

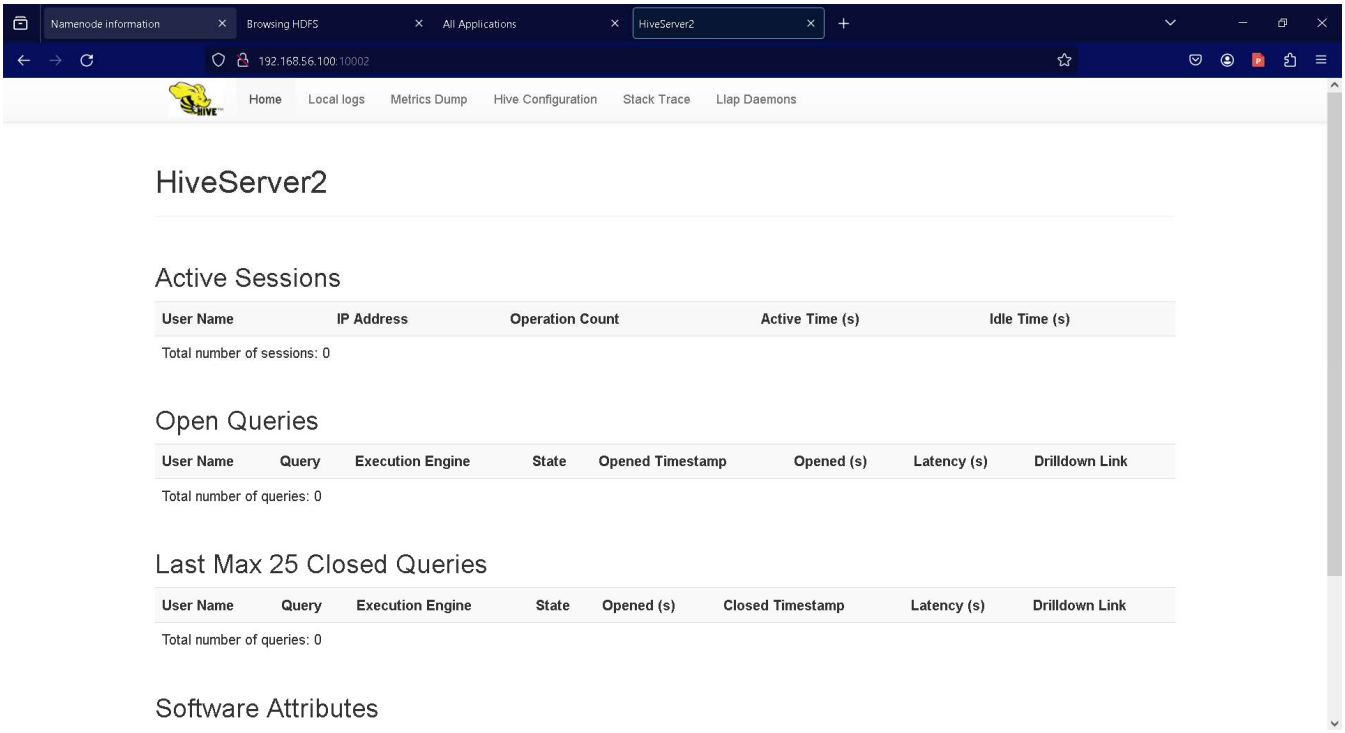
HIVEQL

Basic Commands;

$ schematool -dbType derby -validate

```
hive > SET mapreduce.framework.name=local;
hive> Show databases;
hive>Create database cdacdb;
hive>Create database if not exists hivedb;
hive>Use cdacdb;
hive> show tables;
hive> create table dept ( empno int, empname string, emploc string);
hive> insert into table dept values (1, 'shivaji','maharashtra');
hive> insert into table dept values (2, 'bajirao',' rajasthan');
hive> select * from dept;
```

$ cd /usr/local/hive/conf
$ hive --service hiveserver2 & or use $ hiveserver2
$ ss -ant { verify weather you can view 10000 and 10002 port if yes then use the web browser to access the information }



HIVE DATA TYPES

- **We have primitive and complex datatypes**
- **Numeric types --> INT/INTERGER , SMALLINIT, TINYINT, BIGINIT, FLOAT,DOUBLE, DOUBLE PRESISION, DECIMAL,NUMERIC**
- **Date/time type --> DATE, TIMESTAMP, INTERVAL**
- **String Types ---> STRING, VARCHAR, CHAR**
- **Misc Types ---> BOOLEAN, BINARY**
- **Complex Types --> ARRAYS, MAPS, STRUCTS, UNION**
  - **Arrays are used to store the list of elements**
  - **Maps --> store key/value pairs**
  - **Structs --> for storing parent and child association**
  - **Union --> union types <data_type, data_type, ...>**

Hive Operators

- **Relation operators**
  - **A=B, A==B, A<==>B, A<> B, A !=B, A < B, A <= B, A <= B, A > B, A >= B, A[NOT] BETWEEN B AND C, A IS NULL, A IS NOT NULL, A IS [NOT], A [NOT] LIKE B, A RLIKE B, A REGEXP B**
- **Logical operators**

- **A AND B , A OR B, NOT A, !A , A IN (val1, val2 ...), A NOT IN (val1, val2 ...), [NOT] EXISTS**

- **Arithmetic operators**
  - **A+B (addition) , A-B (subtract), A*B (multiply) , A/B (divide), A DIV B, A % B (modulation / remainder), A & B (AND) , A | B (OR), A ^ B (XOR), ~A (NOT)**

Hive Numeric Types

Below are Numeric Types Hive support and their sizes.

NUMERIC TYPES	DESCRIPTION
TINYINT	1-byte signed integer, from -128 to 127
SMALLINT	2-byte signed integer, from -32,768 to 32,767
INT/INTEGER	4-byte signed integer, from -2,147,483,648 to 2,147,483,647
BIGINT	8-byte signed integer, from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
FLOAT	4-byte single precision floating point number
DOUBLE	8-byte double precision floating point number
DOUBLE PRECISIO N	Alias for DOUBLE, only available starting with Hive 2.2.0
DECIMAL	It accepts a precision of 38 digits.
NUMERIC	Same as DECIMAL type.

Hive Date/Time Types

Below are Hive Date and Timestamp types, these were not available in the initial versions of the Hive and added in later releases. Date type is used to store just Date and Timestamp is used to store both date and time.

Hive provides several Date & Time functions,

you should use these to perform any date & time operations.

DATE/TIME TYPES	DESCRIPTION
TIMESTAMP	Accepts Both Date and Time
DATE	Accepts just Date
INTERVAL	Interval

Hive String Types

Similar to SQL, Hive also supports CHAR and VARCHAR types, and additionally, it also supports STRING type.

Hive provides several String functions,

you should use these to perform any string operations.

STRING TYPES	DESCRIPTION
STRING	The string is an unbounded type. Not required to specify the lenght. It can accept max up to 32,767 bytes.
VARCHAR	Variable length of characters. It is bounded meaning you still need to specify the length like VARCHAR(10).
CHAR	Fixed length of Characters. if you define char(10) and assigning 5 chars, the remaining 5 characters space will be wasted.

Table3 – Hive String Types

Hive Misc Types

MISC TYPES	DESCRIPTION
BOOLEAN	Accepts TRUE/FALSE values
BINARY	Only available starting with Hive 0.8.0

Hive Complex Types

Similar to Spark, Hive also support complex data types which includes Array, Map, Struct and union.
Array is used to store the list of elements. Map is used to store key/value pair. Struct is for parent and child assosiations.

To work with Complex types, you should use Hive Collection Map & Array functions

COMPLEX TYPES	DESCRIPTION
Arrays	ARRAY<data_type>
Maps	MAP<primitive_type, data_type>

Structs	STRUCT<col_name : data_type [COMMENT col_comment], ...>
Union	UNIONTYPE<data_type, data_type, …> Note: Only available starting with Hive 0.7.0.

```
hive> drop database cdacdb;
hive> drop table cdacdb.dept;
hive> drop database cdacdb cascade;
hive> drop database cdacdb restirct;
hive> drop database  if exists cdacdb;
hive> drop database if exists cdacdb cascade;
hive> drop database if exists cdacdb restrict;
hive> describe database cdacdb;
hive> describe schema  cdacdb;
hive> describe database extended cdacdb;
hive> describe schema extended cdacdb;
```

While creating the database it will be present in the hdfs file system - default location is /user/hive/warehouse

We can change the location temporarily to another by specifying the location option while executing the command

```
$ hdfs dfs -mkdir /cdacdir

hive> create database mytemp location '/cdacdir/mytemp'

$ hdfs dfs -ls /cdacdir     {to verify the same}

hive> show tables;
hive> show tables in cdacdb;
Hive >create database emp;
hvie> CREATE TABLE IF NOT EXISTS emp.employee (
        id int,
         name string,
         age int,
        gender string )
        COMMENT 'Employee Table
        ROW FORMAT DELIMITED
        FIELDS TERMINATED BY ',';

hive> use emp;

hive> alter table employee rename to myemp1;

<use insert statement and add few rows to myemp1 table>

hive> truncate table myemp1;

hive> create table emp.my_emp3 like emp.myemp1;

hive>insert into emp.my_emp3 select id,name,age,gender from emp.myemp1 where id IS NOT NULL;


hive> CREATE TABLE IF NOT EXISTS emp.myempl ( id int, name string, age int, gender string )CLUSTERED By (id) into 2 Buckets stored
as ORC TBLPROPERTIES ('transactional' ='true');

{ Exporting data from a table  to a HDFS in csv format }

hive> INSERT OVERWRITE DIRECTORY '/user/hive/warehouse/export'  ROW FORMAT DELIMITED FIELDS TERMINATED BY ','  SELECT *
FROM emp.my_emp3;

Verify it in your hdfs file systems using the following command:-

$ hdfs dfs -cat  /user/hive/warehouse/export/000000_0

Hive> create table emp.my_emp4 like emp.my_emp3;

{ Importing data from HDFS in csv format to a table}
Hive> LOAD DATA INPATH '/user/hive/warehouse/export/000000_0' INTO TABLE my_emp4;

Hive> exit;
```