ISIT312 Big Data Management

Hive Data Structures

Dr Fenghui Ren

School of Computing and Information Technology -
University of Wollongong

Hive Data Structures

Outline

- Primitive Data Types
- Complex Data Types
- Databases
- Tables
- Partitions
- Buckets
- Views

Primitive Data Types

`TINYINT`, 1 byte, example: `10Y`

`SMALLINT`, 2 bytes, example: `10S`

`INT`, 4 bytes, example: `10`

`BIGINT`, 8 bytes, example: `10L`

`FLOAT`, 4 bytes, example: `0.1234567`

`DOUBLE`, 8 bytes, example: `0.1234567891234`

`DECIMAL`, (m,n), example: `3.14`

`BINARY`, n bytes, example: `1011001`

`BOOLEAN`, 1 byte example: `TRUE`

`STRING`, 2G bytes, example: `'Abcdef'`

`CHAR`, 255 bytes, example: `'Hello'`

Primitive Data Types

`VARCHAR`, 1 byte, example: `'Hive'`

`DATE`, YYYY-MM-DD, example: `'2017-05-03'`

`TIMESTAMP`, YYYY-MM-DD HH:MM:SS[.fff...] example: `'2017-05-03 15:10:00.345'`

Hive Data Structures

Outline

- Primitive Data Types
- Complex Data Types
- Databases
- Tables
- Partitions
- Buckets
- Views

Complex Data Types

`ARRAY`: list of values of the same types,

```
example: ['Hadoop', 'Pig','Hive']                    Array type
access: bigdata[1]
```

`MAP`: a set of key-value pairs,

```
example: {'k1':'Hadoop', 'k2':'Pig'}                 Map type
access: bigdata['k2']
```

`STRUCT`: user defined structure of any type of fields,

```
example: {name:'Hadoop', age:24, salary:50000.06}    Struct type
access: bigdata.name
```

Complex Data Types

The following **CREATE TABLE** command creates a table **types** with complex data types columns

A table with columns of types

```
CREATE TABLE types(
   array_col array<string>,
   map_col map<int,string>,
   struct_col struct<a:string, b:int, c:double> );
```

INSERT INTO and **SELECT** statements can be used to load data into a table with complex data types columns

Inserting values into a table

```
INSERT INTO types
   SELECT array('bolt', 'nut', 'screw'),
          map(1,'bolt', 2,'nut', 3,'screw'),
          named_struct('a','bolt', 'b',5, 'c',0.5)
   FROM DUAL;
```

Hive Data Structures

Outline

- Primitive Data Types
- Complex Data Types
- Databases
- Tables
- Partitions
- Buckets
- Views

Databases

Database is a collection of conceptually related tables, i.e. tables that implement a conceptual schema

Database is implemented as a folder/directory in HDFS

A default database is located at `/user/hive/warehouse`

A new database is created in a folder `/user/hive/warehouse`

For example, a database `tpchr` is located at `/user/hive/warehouse/tpchr.db`

Databases

The following `CREATE DATABASE` command creates a database `tpchr`

```
Creating a database

CREATE DATABASE tpchr;
```

To find more information about a database we can use `DESCRIBE DATABASE` command

```
Listing a database

DESCRIBE DATABASE tpchr;
```

A command `USE` makes a database "current" (there is no need to prefix a table name with a database name)

```
Making a database current

USE tpchr;
```

To delete a database we can use `DROP DATABASE` command

```
Dropping a database

DROP DATABASE tpchr;
```

Hive Data Structures

Outline

- Primitive Data Types
- Complex Data Types
- Databases
- Tables
- Partitions
- Buckets
- Views

Tables

An internal table (or managed table) is a table created by Hive in HDFS

If data is already stored in HDFS then an external Hive table can be created to provide a tabular view of the data

Location in HDFS of data stored in an external table is specified in the `LOCATION` properties instead of the default warehouse directory

Hive fully manages the life cycle (add/delete data, create/drop table) of internal tables and data in the internal tables

When an external table is deleted its metadata information is deleted from a metastore and the data is kept in HDFS

Tables

`CREATE TABLE` statement creates an internal table

Creating an internal table

```
CREATE TABLE IF NOT EXISTS intregion(
   R_REGIONKEY DECIMAL(12),
   R_NAME VARCHAR(25),
   R_COMMENT VARCHAR(152) )
      ROW FORMAT DELIMITED FIELDS TERMINATED BY '|'
      STORED AS TEXTFILE;
```

`LOAD DATA` statement loads data into an internal table

Loading data into an internal table

```
LOAD DATA LOCAL INPATH 'region.tbl' INTO TABLE intregion;
```

Tables

`CREATE EXTERNAL TABLE` statement creates an external table

Creating an external table

```
CREATE EXTERNAL TABLE IF NOT EXISTS extregion(
   R_REGIONKEY DECIMAL(12),
   R_NAME VARCHAR(25),
   R_COMMENT VARCHAR(152) )
      ROW FORMAT DELIMITED FIELDS TERMINATED BY '|'
      STORED AS TEXTFILE LOCATION '/user/tpchr/region';
```

`LOAD DATA` statement loads data into an external table

Loading data into an external table

```
LOAD DATA LOCAL INPATH 'region.tbl' INTO TABLE extregion;
```

Tables

An external table can be created "over" an already existing file in HDFS

Loading a file to HDFS

```
hadoop fs -mkdir /user/tpchr/nation
hadoop fs -put nation.tbl /user/tpchr/nation
hadoop fs -ls /user/tpchr/nation
-rw-r--r-- 3 janusz supergroup 401 2017-07-02 10:24 /user/tpchr/nation/nation.tbl
```

Creating an external table over a file in HDFS

```
CREATE EXTERNAL TABLE IF NOT EXISTS extnation(
   N_NATIONKEY DECIMAL(12),
   N_NAME      VARCHAR(25),
   N_COMMENT   VARCHAR(152) )
      ROW FORMAT DELIMITED FIELDS TERMINATED BY '|'
STORED AS TEXTFILE LOCATION '/user/tpchr/nation';
```

Hive Data Structures

Outline

- Primitive Data Types
- Complex Data Types
- Databases
- Tables
- Partitions
- Buckets
- Views

Partitions

To eliminate unnecessary scans of entire table when only a fragment is needed a table can be divided into partitions

A partition corresponds to predefined columns and it is stored as subfolder in HDFS

When a table is searched only required partitions are accessed

Creating a partitioned table

```
CREATE TABLE IF NOT EXISTS part(
   P_PARTKEY DECIMAL(12),
   P_NAME VARCHAR(55),
   P_TYPE VARCHAR(25),
   P_SIZE DECIMAL(12),
   P_COMMENT VARCHAR(23) )
   PARTITIONED BY (P_BRAND VARCHAR(20))
      ROW FORMAT DELIMITED FIELDS TERMINATED BY '|'
      STORED AS TEXTFILE;
```

Partitions

A partition must be added before data is loaded

```
Adding a partition

ALTER TABLE part ADD PARTITION (P_BRAND='GoldenBolts');
```

```
Listing partitions

show partitions part;
OK
p_brand=GoldenBolts
Time taken: 0.072 seconds, Fetched: 1 row(s)
```

A command that loads a file into a table can be used to load a partition

```
Loading data into a partition

LOAD DATA LOCAL INPATH '/local/home/janusz/HIVE-EXAMPLES/TPCHR/part.txt'
OVERWRITE INTO TABLE part PARTITION (P_BRAND='GoldenBolts');
```

A partition is stored in HDFS as a subfolder

```
Finding a partition in HDFS

hadoop fs -ls /user/hive/warehouse/part
Found 1 items
drwxrwxr-x - janusz supergroup 0 2017-07-01
19:00 /user/hive/warehouse/part/p_brand=GoldenBolts
```

Hive Data Structures

Outline

Primitive Data Types

Complex Data Types

Databases

Tables

Partitions

Buckets

Views

Buckets

Another way to speed up processing of a table is to divide it into buckets

A bucket corresponds to segment of file in HDFS

The values in a bucket column will be hashed by a user defined number into buckets.

Creating a table with buckets

```
CREATE TABLE customer(
   C_CUSTKEY DECIMAL(12),
   C_NAME VARCHAR(25),
   C_PHONE CHAR(15),
   C_ACCTBAL DECIMAL(12,2) )
   CLUSTERED BY (C_CUSTKEY) INTO 2 BUCKETS
      ROW FORMAT DELIMITED FIELDS TERMINATED BY '|';
```

Setting MapReduce and Hive parameters

```
set map.reduce.tasks = 2;
set hive.enforce.bucketing = true;
```

Buckets

INSERT can be used to populate a bucket table

Inserting into a table with buckets

```
INSERT INTO customer
values(1,'Customer#000000001','25-989-741-2988',711.56);
INSERT INTO customer
values(2,'Customer#000000002','23-768-687-3665',121.65);
INSERT INTO customer
values(3,'Customer#000000003','11-719-748-3364',7498.12);
INSERT INTO customer
values(4,'Customer#000000004','14-128-190-5944',2866.83);
INSERT INTO customer
values(5,'Customer#000000005','13-750-942-6364',794.47)
```

Hive Data Structures

Outline

Primitive Data Types

Complex Data Types

Databases

Tables

Partitions

Buckets

Views

Views

Views are logical data structures that simplify queries

Views do not store data or get materialized

Once a views is created its definition is frozen and changes in the tables used in the view definition are not reflected in the view schema

Creating a view

```
CREATE VIEW vcustomer AS
   SELECT C_CUSTKEY, C_NAME, C_PHONE
   FROM CUSTOMER
   WHERE C_CUSTKEY < 5;
```

References

Gross C., GuptaA., Shaw S., Vermeulen A. F., Kjerrumgaar D., Practical Hive: A guide to Hadoop's Data Warehouse System, Apress 2016, Chapter 4 (Available through UOW library)

Lee D., Instant Apache Hive essentials how-to: leverage your knowledge of SQL to easily write distributed data processing applications on Hadoop using Apache Hive, Packt Publishing Ltd. 2013 (Available through UOW library)

Apache Hive