A NoSQL Database - Hive

Dania Abed Rabbou

Hive

- A data-warehousing framework built on top of Hadoop by Facebook

- Grew from a need to analyze huge volumes of daily data traffic (~10 TB) generated by Facebook

- Facebook owns the second largest Hadoop cluster in the world (~2 PB)

Hadoop & Hive Usage at Facebook

- To produce daily and hourly summaries such as reports on the growth of users, page views, average time spend on different pages etc.
- To perform backend processing for site features such as people you may like and applications you may like.
- To quantify the success of advertisement campaigns and products.
- To maintain the integrity of the website and detect suspicious activity.

Hive vs. RDBMs

1. **Schema on read:**

Traditionally the table's schema is enforced at data load time (schema on write). Hive enforces it at query time (a load operation is simply a quick file move)

2. **Updates:**

Table updates are only possible by transforming all the data into a new table (i.e. no appends)

3. **Transactions :**

Hive does not support concurrent accesses to tables and hence application-level concurrency and locking mechanisms are needed.

4. **Indexes:**

Support provided but relatively immature

HiveQL: Hive's SQL Dialect

- HiveQL adopts a SQL-like syntax
- HiveQL supports the following datatypes:

Primitive:	**Complex:**

TINYINT (1 byte),
SMALLINT (2 bytes),
INT (4 bytes),
BIGINT (8 bytes),
DOUBLE,
BOOLEAN,
STRING
	ARRAY, MAP, STRUCT
	
	Eg: CREATE TABLE tbl (
	    col1 ARRAY<INT>,
	    col2 MAP<STRING, INT>,
	    col3 STRUCT<a:STRING, b:INT, c:DOUBLE>
	  );

Hue: Hadoop's Web Interface

- Hue is an open-source user-friendly web-interface for Hadoop components (including HDFS, Hive, Pig, etc.)

- Browse to your Hue interface located at:

  <andrew_id>-hdp.qatar.cmu.local:8000

  username: hue

  password: SummerYet

Loading Data into HDFS

- Any datasets needed for loading into tables must be moved to HDFS
- Load some test datasets into HDFS:
  - Navigate to the File Browser
  - Create a new directory, say DatasetsSource
  - Move into DatasetsSource and upload three csv files namely customer_details, recharge_details, and customer_details_with_addresses

Creating Databases

- To create a new Hive database:
  - Browse to Beewax (Hive’s UI)
  - Click on the Databases tab
  - Create a new database, say Customers

Creating Tables

- Create two tables under the database Customers:
  - In Beewax, click on the Query Editor tab
  - Create tables customer_details & recharge_details

```
CREATE TABLE IF NOT EXISTS
customer_details
(phone_num STRING,
plan STRING,
date STRING,
status STRING,
balance STRING,
region STRING)
COMMENT "Customer Details"
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ","
STORED AS TEXTFILE;
```

```
CREATE TABLE IF NOT EXISTS
recharge_details
(phone_num STRING,
date STRING,
channel STRING,
plan STRING,
amount STRING)
COMMENT "Recharge Details"
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ","
STORED AS TEXTFILE;
```

Loading Data from HDFS to Tables

- Load data into the two tables:
  - In the Query Editor tab, load each dataset previously uploaded to HDFS into its respective table

```
LOAD DATA INPATH "/user/hue/DatasetsSource/customer_details.csv"
OVERWRITE INTO TABLE customer_details;
```

```
LOAD DATA INPATH "/user/hue/DatasetsSource/recharge_details.csv"
OVERWRITE INTO TABLE recharge_details;
```

Browse back to /user/hue/DatasetsSource; the datasets loaded into tables **disappeared**!!

Hive moves the datasets to a default warehousing folder

Deleting Tables

- To delete a table:
  - In the Tables tab, choose a table to delete and click drop

The table including its metadata and *data* is **deleted**!

In other words, the loaded data no longer exists anywhere!

Creating External Tables

- To control the creation and deletion of data, use external tables

```
CREATE EXTERNAL TABLE IF NOT EXISTS
customer_details
(phone_num STRING,
plan STRING,
date STRING,
status STRING,
balance STRING,
region STRING)
COMMENT "Customer Details"
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ","
STORED AS TEXTFILE
LOCATION "/user/hue/LoadedDatasets";
```

A path were the loaded dataset will be stored. If the table is deleted, the data stays around

Displaying Data in Tables

- Consider the schemas of our two tables:

  customer_details(phone_num, plan, date, status, balance, region)

  recharge_details(phone_num, date, channel, plan, amount)

- Display the records in customer_details

SQL:
```
SELECT * FROM customer_details;
```

HiveQL:
```
SELECT * FROM customer_details;
```

Updating Tables

- Consider the schemas of our two tables:

  customer_details(phone_num, plan, date, status, balance, region)

  recharge_details(phone_num, date, channel, plan, amount)

- Let's update plan 4060 to a recharge amount of 500

SQL:

```
UPDATE recharge_details
SET amount=500
WHERE plan=4060;
```

HiveQL:

```
INSERT OVERWRITE TABLE recharge_details
SELECT phone_num, date, channel, plan,
CASE WHEN plan=4060 THEN 500 ELSE amount END as amount
FROM recharge_details;
```

The entire table contents is re-written

Joining Tables

- Consider the schemas of our two tables:

  customer_details(phone_num, plan, date, status, balance, region)

  recharge_details(phone_num, date, channel, plan, amount)

- Let’s display the recharge amount per customer

SQL:
```
SELECT c.phone_num, r.amount
FROM customers_details c, recharge_details r
WHERE c.phone_num = r.phone_num;
```

HiveQL:
```
SELECT customer_details.phone_num, recharge_details.amount
FROM customer_details JOIN recharge_details ON
(customer_details.phone_num = recharge_details.phone_num);
```

Tables with Complex Datatypes

- Let's add a new field to the customer_details table called "addresses". This field shall hold a list of addresses per customer

```
CREATE TABLE IF NOT EXISTS
customer_details_2
(phone_num STRING,
plan STRING,
date STRING,
status STRING,
balance STRING,
region STRING
addresses ARRAY<STRING>)
COMMENT "Customer Details"
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ","
COLLECTION ITEMS TERMINATED BY “;"
STORED AS TEXTFILE;
```

```
LOAD DATA INPATH
"/user/hue/DatasetsSource/
customer_details_with_addresses.csv"
OVERWRITE INTO TABLE customer_details_2;
```

```
SELECT * FROM customer_details_2;
```

```
SELECT addresses[0] FROM customer_details_2;
```

Built-In Functions

- Hive provides many built-in functions.

  To list them all:

```
SHOW FUNCTIONS;
```

- To understand the functionality of a function:

```
DESCRIBE FUNCTION array_contains;
```

- Let’s display those customer records whose addresses include ‘Qatar’

```
SELECT * FROM customer_details_2
WHERE array_contains(addresses, "Oman");
```

Hive’s Additional Features

- Allows User-Defined Functions (UDFs). UDFs can be written in Java and integrated with Hive.

- Support a new construct (TRANSFORM .. USING ..) to invoke an external script or program.
  - Hive ships invokes the specified program, feeds it data, and reads data back.
  - Useful for pre-processing datasets before loading them into tables etc.