Big Data Technologies

Agenda

- MR: Shuffle
- Hive

Shuffle

- Output of mappers is shuffled across multiple reducers.
- Output of each mapper is divided into partitions (equals to number of reducers). These partitions are copied/collected on respective reducers.

Data warehousing

- Data warehouse for handling huge amount of structured data/tabular data.
- Two types
  - OLAP -- Online Analytical Processing
  - OLTP -- Online Processing Processing
- OLAP
  - Typical RBDMS take long time for executing analytical queries on huge data.
  - OLAP DWH is designed for executing analytical workload in fastest possible time.
  - e.g. Hive, Oracle BI, ...
- OLTP
  - Operational database in which live business data is processed (DML).
  - OTLP is done with traditional RDBMS e.g. Oracle, MS-SQL, MySQL, Postgre-SQL, ...
- Data warehousing is a process by which we can produce analytical reports as fast as possible. For this we might need to reorganize data into different table structures, de-normalize the database, use collection types, partitioning the data, etc.

Code share

- http://172.18.4.63:4200

Hive

- Hive is DWH on Hadoop.
- https://cwiki.apache.org/confluence/display/hive/languagemanual+ddl
- https://cwiki.apache.org/confluence/display/hive/languagemanual+udf

Hive introduction

- terminal> beeline -u jdbc:hive2://localhost:10000/default -n $USER

```
SHOW DATABASES;

CREATE DATABASE dbda;

USE dbda;

SHOW TABLES;

CREATE TABLE students(id INT, name CHAR(20), marks DOUBLE)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

SHOW TABLES;

INSERT INTO students VALUES (1, 'Soham', 95.45);

INSERT INTO students VALUES
(2, 'Prisha', 98.87),
(3, 'Sakshi', 96.31),
(4, 'Madhura', 96.23);

SELECT * FROM students
LIMIT 10;
```

```
LOAD DATA LOCAL
INPATH '/home/nilesh/sep22/dbda/bigdata/day08/newstudents.csv'
INTO TABLE students;

SELECT * FROM students
LIMIT 10;

!quit
```

Hive types

- terminal> beeline -u jdbc:hive2://localhost:10000/dbda -n $USER

```
SHOW TABLES;

CREATE TABLE contacts(
id INT,
name STRING,
emails ARRAY<STRING>,
addr STRUCT<area:STRING,dist:STRING,pin:INT>,
phone MAP<STRING,STRING>
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
COLLECTION ITEMS TERMINATED BY '|'
MAP KEYS TERMINATED BY ':'
STORED AS TEXTFILE;

LOAD DATA LOCAL
INPATH '/home/nilesh/sep22/dbda/bigdata/data/contacts.csv'
INTO TABLE contacts;

DESCRIBE contacts;
```

```
SELECT * FROM contacts
LIMIT 5;

SELECT name, emails[0], addr.dist, phone['mobile'] FROM contacts
LIMIT 5;

-- list people with email -- ghule.nilesh@gmail.com

SELECT name, emails FROM contacts
WHERE emails[1]='ghule.nilesh@gmail.com'
LIMIT 5;

SELECT name, emails FROM contacts
WHERE ARRAY_CONTAINS(emails,'ghule.nilesh@gmail.com')
LIMIT 5;

-- list people with multiple emails
SELECT name, emails FROM contacts
WHERE SIZE(emails) > 1;

-- list people from pune
SELECT name, emails FROM contacts
WHERE addr.dist = 'pune';

-- find a person with mobile number 9881208114
SELECT name, phone FROM contacts
WHERE phone['mobile']='9881208114';

-- find number of people per dist
SELECT addr.dist AS dist, COUNT(id) AS cnt FROM contacts
GROUP BY addr.dist;
```

CREATE & DROP TABLE

- Limitations of FIELDS TERMINATED BY in CSV files

```
CREATE TABLE movies(
id INT,
title STRING,
genres STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;

DESCRIBE movies;

LOAD DATA LOCAL
INPATH '/tmp/movies/movies.csv'
INTO TABLE movies;

SELECT * FROM movies
LIMIT 20;

DROP TABLE movies;

CREATE TABLE movies(
id INT,
title STRING,
genres STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '^'
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;

LOAD DATA LOCAL
INPATH '/tmp/movies/movies_caret.csv'
INTO TABLE movies;
```

```
SELECT * FROM movies
LIMIT 20;

DROP TABLE movies;
```

SerDe

- CSV file format
  - fields seperated by ','.
  - if field contains ',' then field value is in double-quotes.
  - if field contains '"' then escape it using ''
- SerDe = Serializer + Deserializer

```
CREATE TABLE movies_staging(
id INT,
title STRING,
genres STRING
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES (
'separatorChar' = ',',
'quoteChar'     = '\"',
'escapeChar'    = '\\'
)
TBLPROPERTIES('skip.header.line.count'='1');

DESCRIBE movies_staging;

LOAD DATA LOCAL
INPATH '/tmp/movies/movies.csv'
INTO TABLE movies_staging;
```

```
SELECT * FROM movies_staging
LIMIT 20;
```

```
CREATE TABLE ncdc_staging(
yr SMALLINT,
temp INT,
quality TINYINT
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES(
'input.regex'='^.{15}([0-9]{4}).{68}([-\+][0-9]{4})([0-9]{1}).*$'
);

LOAD DATA LOCAL
INPATH '/home/nilesh/sep22/dbda/bigdata/data/ncdc/*'
INTO TABLE ncdc_staging;

SELECT * FROM ncdc_staging
LIMIT 10;

SELECT yr, AVG(temp) AS avgtemp FROM ncdc_staging
WHERE temp != 9999 AND quality IN (0,1,2,4,5,9)
GROUP BY yr;

SELECT yr, AVG(temp) AS avgtemp FROM ncdc_staging
WHERE temp != 9999 AND quality IN (0,1,2,4,5,9)
GROUP BY yr
ORDER BY 2 DESC
LIMIT 1;

SELECT yr, AVG(temp) AS avgtemp FROM ncdc_staging
WHERE temp != 9999 AND quality IN (0,1,2,4,5,9)
GROUP BY yr
ORDER BY 2 ASC
```

```
LIMIT 1;

(SELECT 'Coolest' AS category, yr, AVG(temp) AS avgtemp FROM ncdc_staging
WHERE temp != 9999 AND quality IN (0,1,2,4,5,9)
GROUP BY yr
ORDER BY 3 ASC
LIMIT 1)
UNION
(SELECT 'Hotest' AS category, yr, AVG(temp) AS avgtemp FROM ncdc_staging
WHERE temp != 9999 AND quality IN (0,1,2,4,5,9)
GROUP BY yr
ORDER BY 3 DESC
LIMIT 1);
```

Store output in Table

```
CREATE TABLE ncdc_avgtemp AS
SELECT yr, AVG(temp) AS avgtemp FROM ncdc_staging
WHERE temp != 9999 AND quality IN (0,1,2,4,5,9)
GROUP BY yr;

DESCRIBE ncdc_avgtemp;

SELECT * FROM ncdc_avgtemp
LIMIT 20;

(SELECT 'Coolest' AS category, yr, avgtemp FROM ncdc_avgtemp
ORDER BY avgtemp ASC
LIMIT 1)
UNION
(SELECT 'Hotest' AS category, yr, avgtemp FROM ncdc_avgtemp
ORDER BY avgtemp DESC
LIMIT 1);
```

Hive Views

```
CREATE VIEW v_ncdc_avgtemp AS
SELECT yr, AVG(temp) AS avgtemp FROM ncdc_staging
WHERE temp != 9999 AND quality IN (0,1,2,4,5,9)
GROUP BY yr;

SELECT * FROM v_ncdc_avgtemp
LIMIT 20;

(SELECT 'Coolest' AS category, yr, avgtemp FROM v_ncdc_avgtemp
ORDER BY avgtemp ASC
LIMIT 1)
UNION
(SELECT 'Hotest' AS category, yr, avgtemp FROM v_ncdc_avgtemp
ORDER BY avgtemp DESC
LIMIT 1);

SHOW VIEWS;

SHOW TABLES;

!quit
```