Apache Hive

Dr. Lee

Learning Goals

- Describe what Hive is and when to use it
- Describe the data pipeline and use cases
- Describe how Hive fits in the Hadoop ecosystem
- Describe data types in Hive

Too Much Data?

You work in a large climate research center. You

- already know SQL
- have accumulated petabytes of weather measurements
- migrated data to HDFS

How do you process this data?

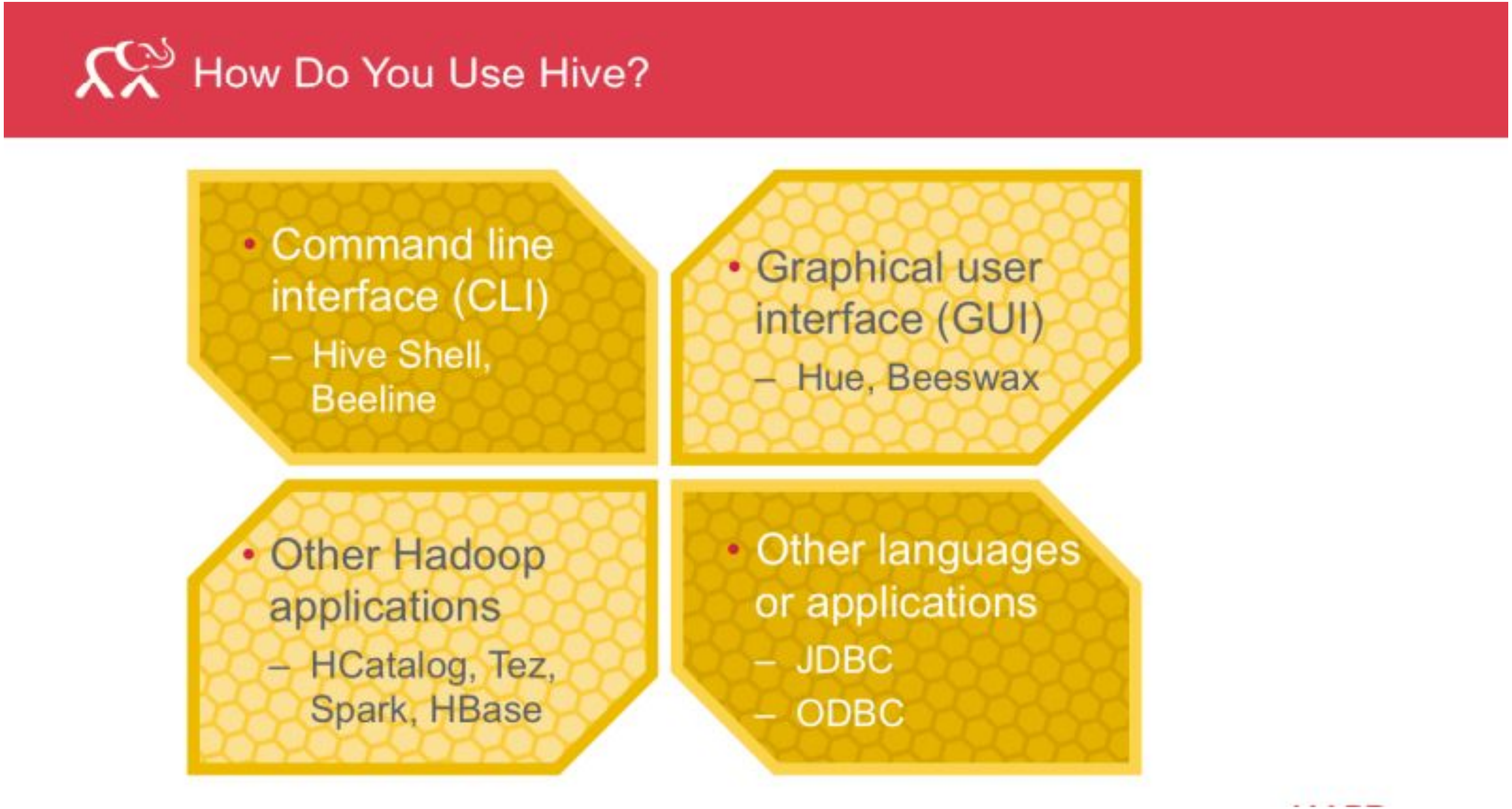
What are Hive & HiveQL?

Hive is

- a data warehouse application
- part of the Hadoop ecosystem

Hive Query Language is

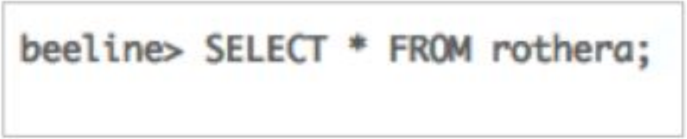
- called HiveQL (HQL)
- A SQL-like language used to explore data in HDFS



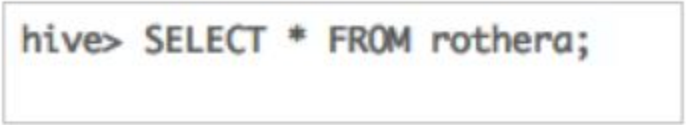
How Do You Use Hive?

- Command line interface (CLI)
  - Hive Shell, Beeline
- Graphical user interface (GUI)
  - Hue, Beeswax
- Other Hadoop applications
  - HCatalog, Tez, Spark, HBase
- Other languages or applications
  - JDBC
  - ODBC

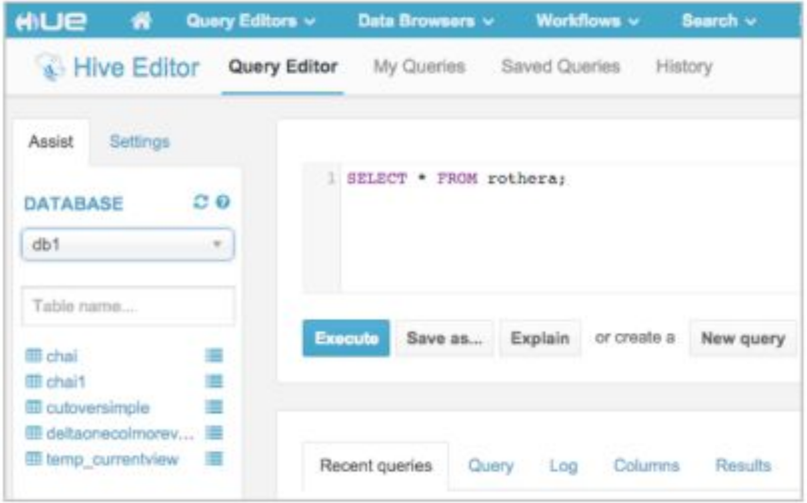
CLI vs GUI



```
beeline> SELECT * FROM rothera;
```



```
hive> SELECT * FROM rothera;
```



```
SELECT * FROM rothera;
```

When should you use Hive?

- Easy to learn if you already know SQL
- Widely used in the Hadoop ecosystem
- Good version compatibility
- Query
  - large amounts of data
  - structured data
  - in batches

When to use SQL or HQL?

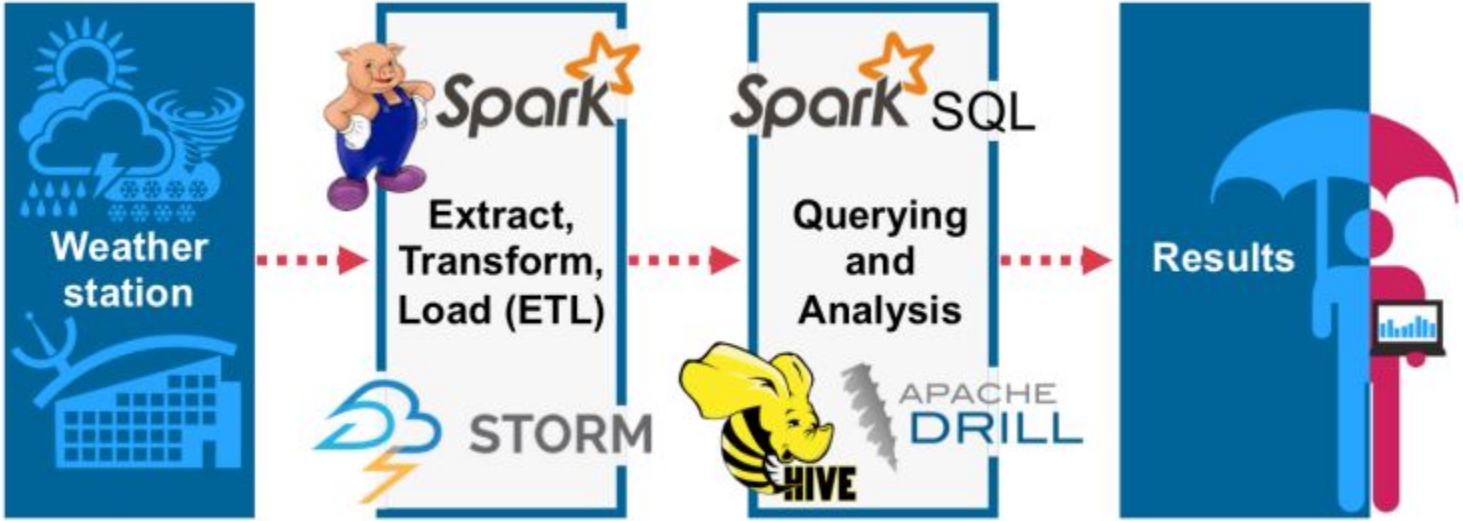
SQL	HQL
Works on RDBMS	Works on HDFS
Interactive; queries provide results in near real time	Batch; queries have overhead due to MapReduce
Works best on small or medium datasets (megabytes or gigabytes)	Works best on large datasets (terabytes or petabytes)

Knowledge Check

Who should be familiar with Hive? Check all that apply.

a) Data scientists working with data on an HDFS
b) Business analysts working with data on an RDBMS
c) Data analysts who want interactive, real time queries
d) Hadoop developers who work with data scientists

The Data Pipeline



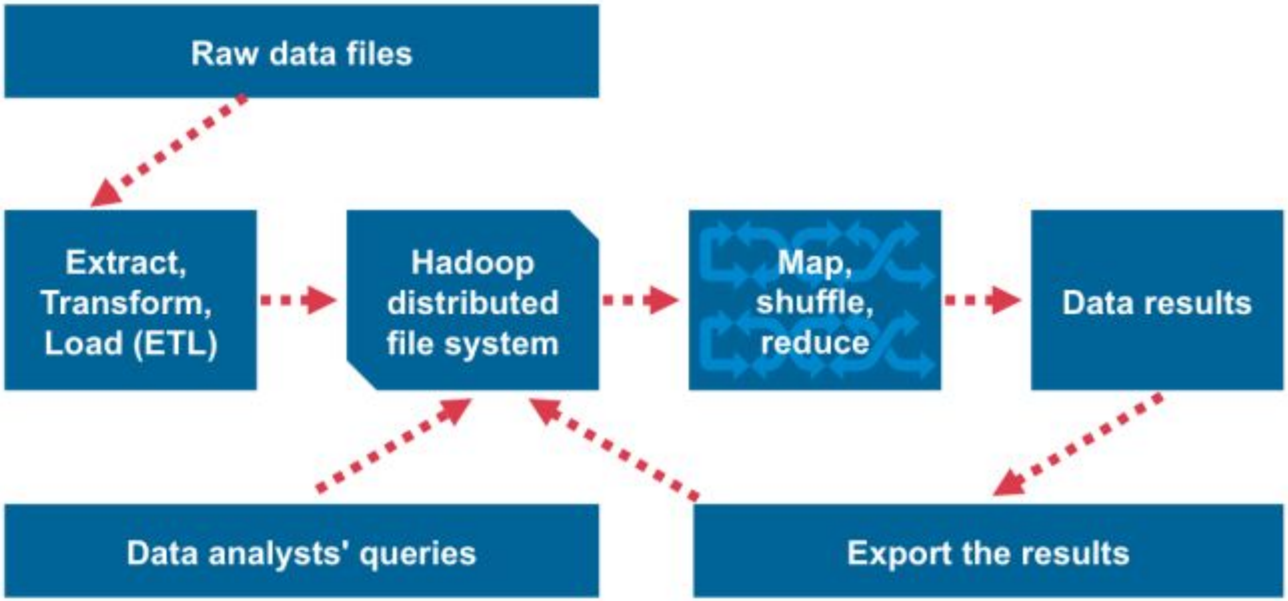
Weather station

Extract, Transform, Load (ETL)

Querying and Analysis

Results

MapReduce and the Data Pipeline



Raw data files

Extract, Transform, Load (ETL)

Hadoop distributed file system

Map, shuffle, reduce

Data results

Data analysts' queries

Export the results

Discussion – How do you interact with data?

Think about the data you work with, as a developer or data analyst. What part of the data pipeline are you usually involved with? How do you think you might use Hive in your work?

Hive & other SQL-on-Hadoop tools

- First SQL-on-Hadoop tool
- New tools like Drill:
  - often faster than Hive
  - may lack key functions
  - may suffer compatibility issues
- Familiarity with SQL & Hive helps you learn new tools

When to use Hive or Pig?

Hive (and HiveQL)	Pig (and Pig Latin)
Declarative language	Procedural language
Used mostly by data analysts	Used by both data analysts and developers
Used to run batch queries on structured data	Used to automate ETL for unstructured data

HiveQL & MapReduce

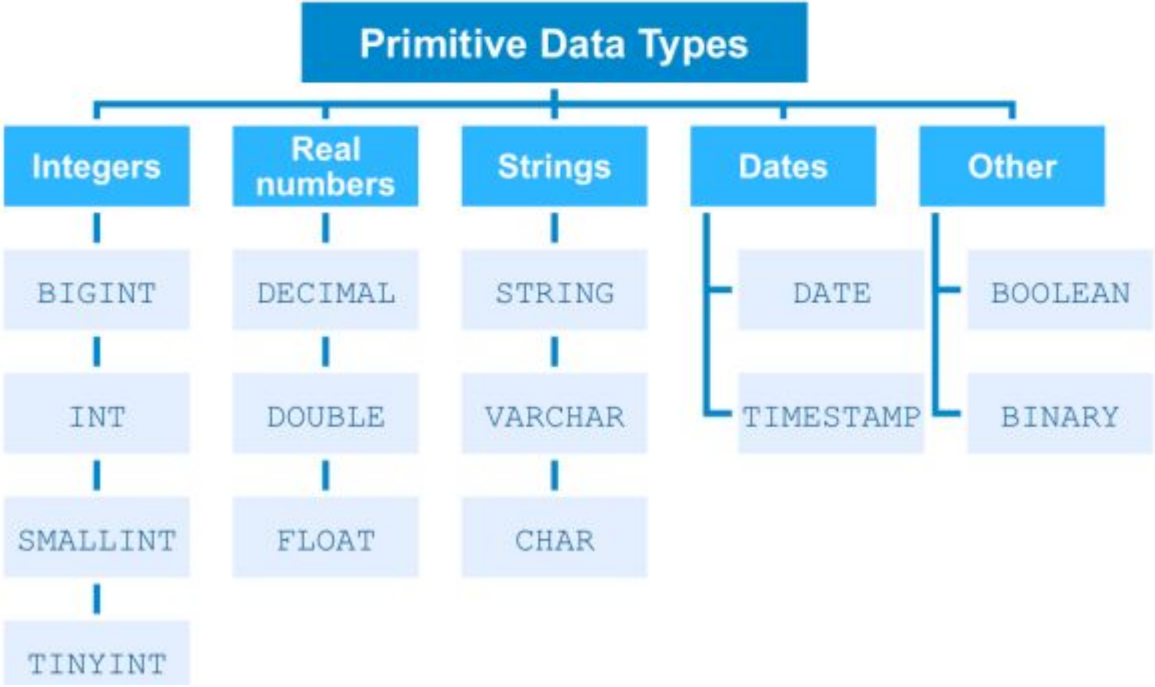
Hive queries

- work on HDFS
- sent in batches
- Queries sent from Beeline or Hive CL
- Most queries trigger MapReduce

Options for optimization of Hive

- Skip reads
  - Partition, ORC
- Minimize shuffle
  - Bucket, Sort
- Hive on Spark
- Hive on Tez

Primitive data types

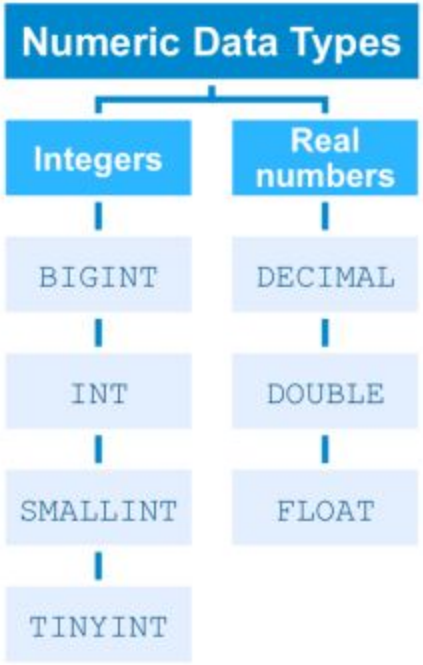


Primitive Data Types

- Integers
  - BIGINT
  - INT
  - SMALLINT
  - TINYINT
- Real numbers
  - DECIMAL
  - DOUBLE
  - FLOAT
- Strings
  - STRING
  - VARCHAR
  - CHAR
- Dates
  - DATE
  - TIMESTAMP
- Other
  - BOOLEAN
  - BINARY

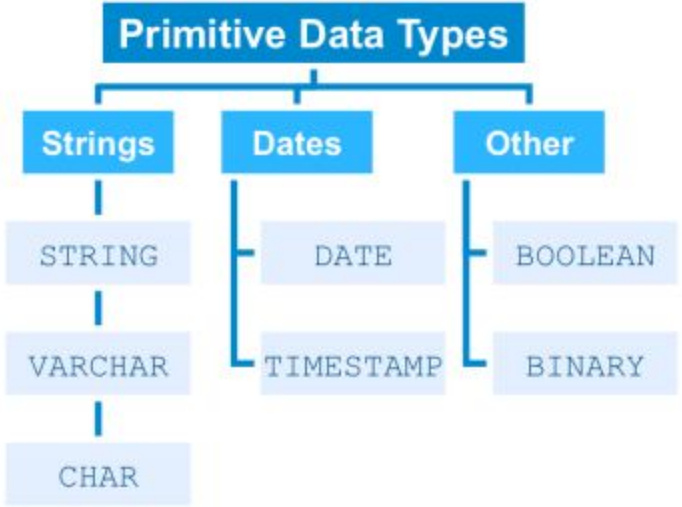
Numeric data

- Integers are signed whole numbers
  - `TINYINT` -2^{7} to $2^{7}-1$
  - `SMALLINT` -2^{15} to $2^{15}-1$
  - `INT` -2^{31} to $2^{31}-1$
  - `BIGINT` -2^{63} to $2^{63}-1$
- Real numbers include decimals
  - `FLOAT` 4 bytes
  - `DOUBLE` 8 bytes
  - `DECIMAL` 32 bytes
- **`NUMERIC`**, **`MONEY`** datatypes from SQL are not available in HiveQL



Other primitive data types

- Strings
  - `'Hello world!'`
- Dates
  - `DATE 1970-01-24`
  - `TIMESTAMP 1970-01-24 08:52:48.123`
- Other
  - `TRUE or FALSE`
  - Array of bytes



Primitive Data Types

Strings

STRING

VARCHAR

CHAR

Dates

DATE

TIMESTAMP

Other

BOOLEAN

BINARY

Complex data types

- **ARRAY<datatype>**
  - stationNames ARRAY<STRING>
  - {"rothera", "airport", "mountain"}
- **MAP<primitive,datatype>**
  - stationIDs MAP<INT,STRING>
  - {201:"airport", 403:"mountain"}
- **STRUCT<colname:datatype,...>**
  - stationLocation STRUCT<name:STRING, longitude:INT, latitude:INT>
  - {name:"rothera", longitude:67, latitude:68}