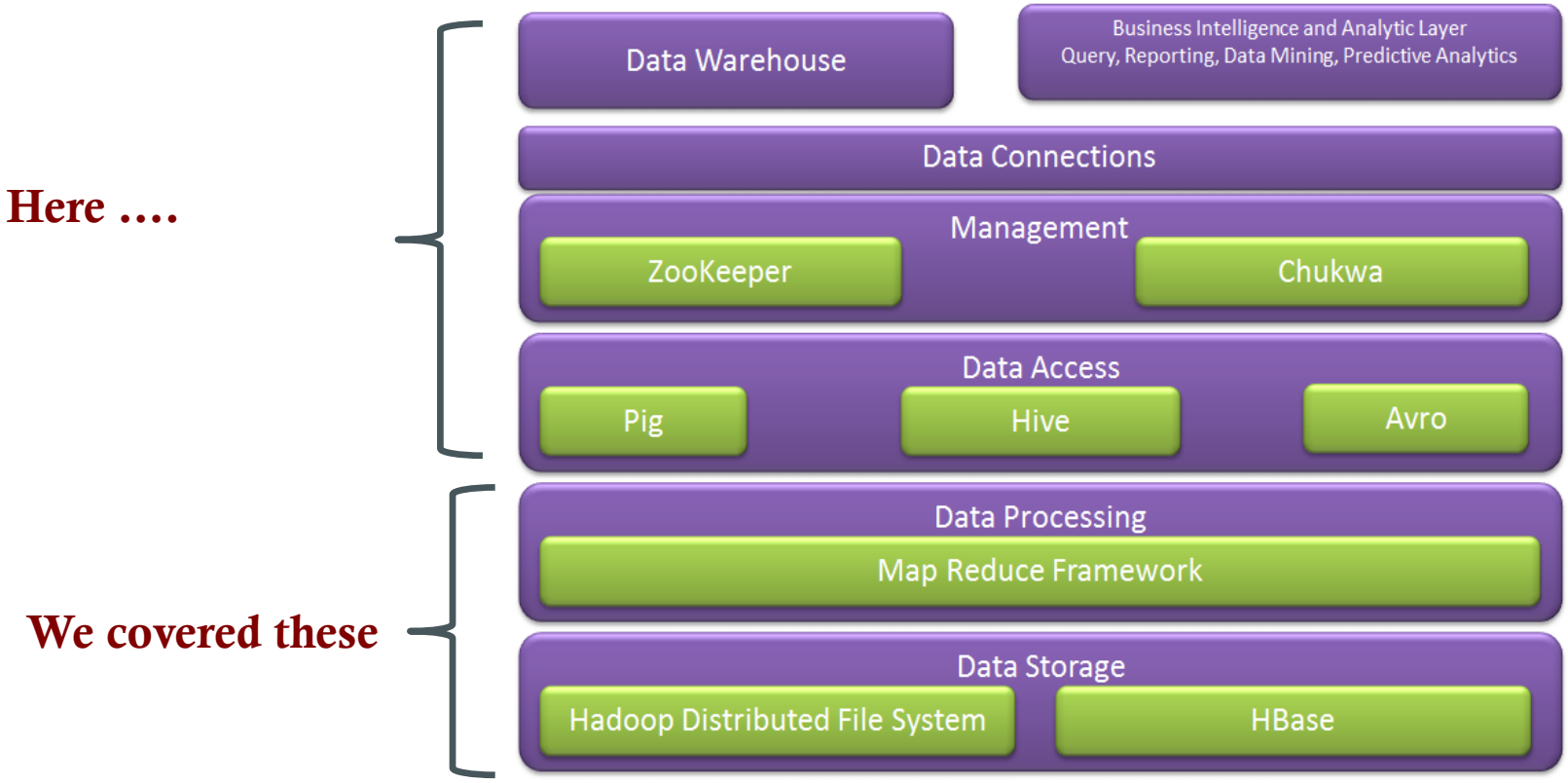
MapReduce High-Level Languages

HIVE (briefly)

Hadoop Ecosystem



Motivation

- Yahoo worked on Pig to facilitate application deployment on Hadoop.
  - **Their focus was unstructured data**

- Facebook worked on Hive for deploying warehouse solutions on Hadoop.
  - **Their focus was structured data**

Apache Hive

- Data warehouse infrastructure on top of Hadoop for providing data summarization, query, and analysis
- **Hive Provides**
  - ETL: Extract-Transform-Load
  - Structure
  - Access to different storage (HDFS or HBase)
  - Query execution via MapReduce
- **Key Building Principles**
  - SQL is a familiar language
  - Extensibility – Types, Functions, Formats, Scripts
  - Performance

Hive deals with Structured Data

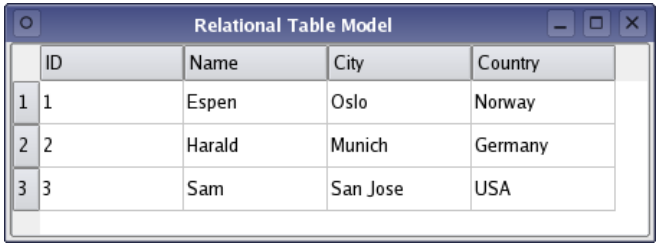
- Data Units:
  - Databases
  - Tables
  - Partitions
  - Buckets (or clusters)

Structure is known inadvance

Very similar to SQL and Relational DBs

Hive vs. DB

DB



	ID	Name	City	Country
1	1	Espen	Oslo	Norway
2	2	Harald	Munich	Germany
3	3	Sam	San Jose	USA

Data is parsed at insertion time

Data types are checked

Loaded into relational table

Hive

No parsing at insertion time

Data remain in its file

File is inserted into HDFS directory corresponding to the table

Hive DDL Commands

- **CREATE TABLE** sample (foo INT, bar STRING) **PARTITIONED BY** (ds STRING);
- **SHOW TABLES** '.*s';
- **DESCRIBE** sample;
- **ALTER TABLE** sample **ADD COLUMNS** (ne
- **DROP TABLE** sample;

A table in Hive is HDFS directory in Hadoop

```
create table table_name (
  id                int,
  dtDontQuery       string,
  name              string
)
partitioned by (date string)
```

Schema is known at creation time (like DB schema)

Partitioned tables have “sub-directories”, one for each partition

Hive DML

Load data from local file system

Delete previous data from that table

- **LOAD DATA LOCAL INPATH** ./sample.txt' **OVERWRITE INTO TABLE** sample;

Load data from HDFS

Augment to the existing data

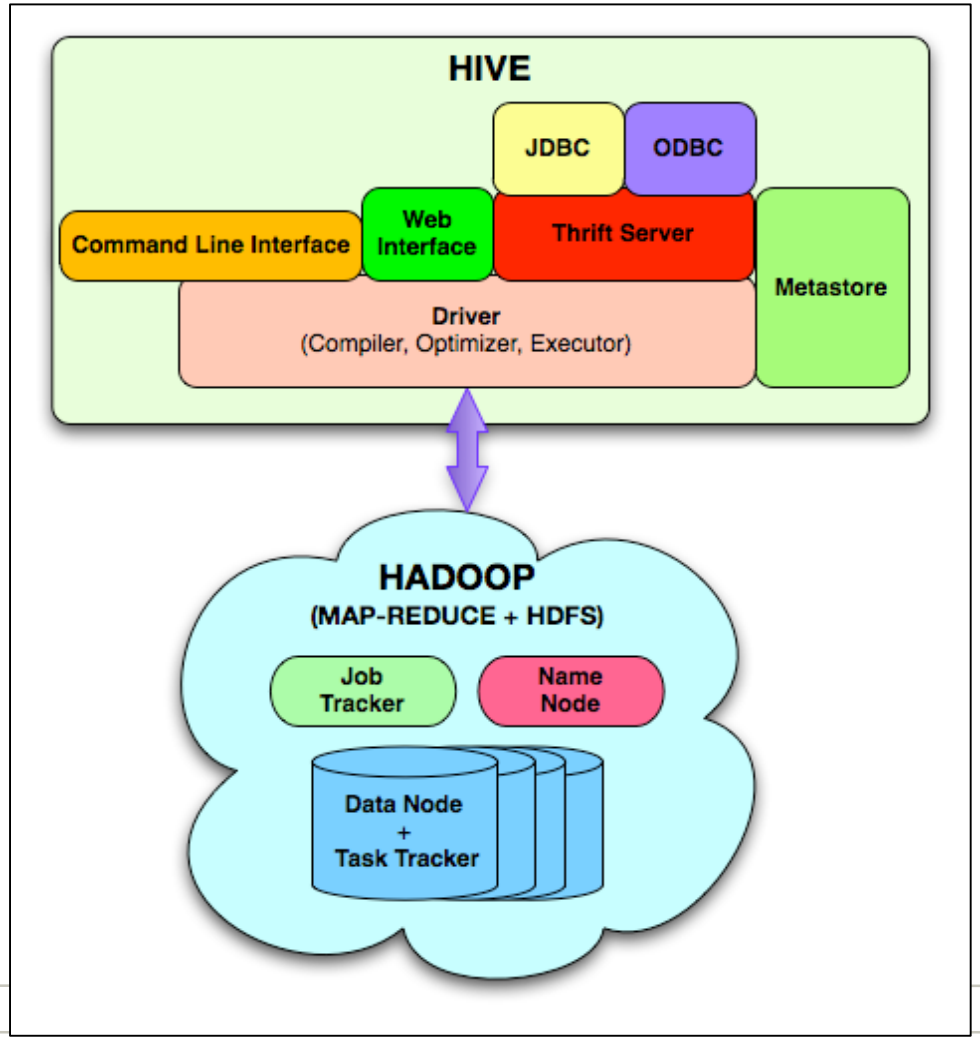
- **LOAD DATA INPATH** '/user/falvariz/hive/sample.txt' **INTO TABLE** partitioned_sample **PARTITION** (ds='2012-02-24');

Must define a specific partition for partitioned tables

Loaded data are files copied to HDFS under the corresponding directory

Hive Components

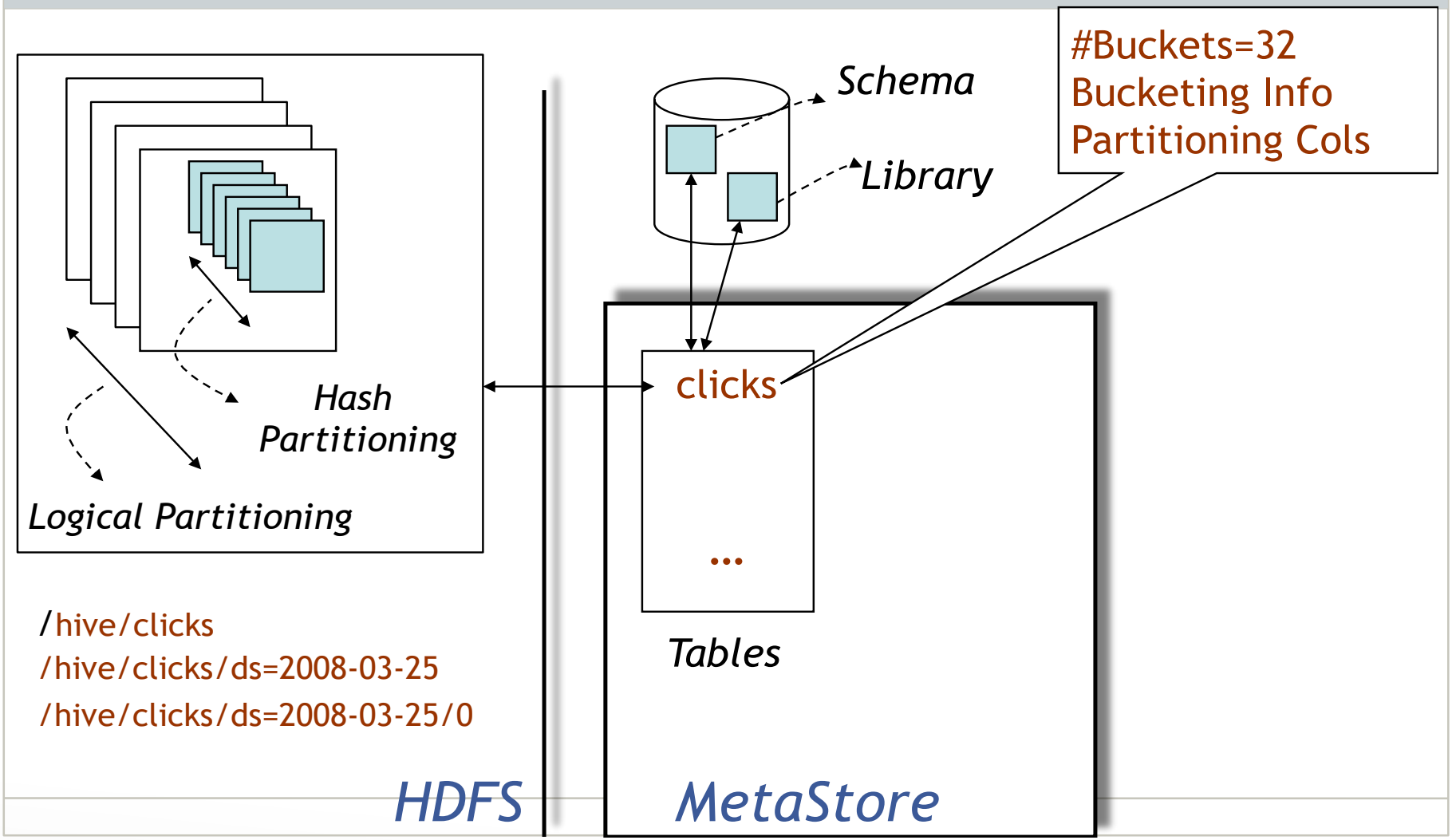
- **Hive CLI:** Hive Command Line Interface
- **MetaStore:** For storing the schema information, data types, partitioning columns, etc…
- **Hive QL:** The query language, compiler, and executer
- **Thrift Server:** cross-language framework to support many languages C, Java, Python, Ruby, PHP



Data Model

- **3-Levels: Tables → Partitions → Buckets**
- **Table: maps to a HDFS directory**
  - Table R: Users all over the world
- **Partition: maps to sub-directories under the table**
  - Partition R by country name
  - It is the user's responsibility to upload the right data to the right partition
- **Bucket: maps to files under each partition**
  - Divide a partition into buckets based on a hash function on a certain column(s)

Data Model


#Buckets=32
Bucketing Info
Partitioning Cols
Schema
Library
Hash
Partitioning
Logical Partitioning
clicks
...
Tables
/hive/clicks
/hive/clicks/ds=2008-03-25
/hive/clicks/ds=2008-03-25/0
HDFS
MetaStore

Queries

- **SELECT MAX**(foo) **FROM** sample;
- **SELECT** ds, **COUNT**(*), **SUM**(foo) **FROM** sample **GROUP BY** ds;

Hive allows the From clause to come first !!!

Store the results into a table

- **FROM** sample s **INSERT OVERWRITE TABLE** bar **SELECT** s.bar, count(*) WHERE s.foo > 0 **GROUP BY** s.bar;
- **SELECT** * **FROM** customer c **JOIN** order_cust o **ON** (c.id=o.cus_id);

Query Examples III: Multi-Insertion

```
FROM page_view_stg pvs

  INSERT OVERWRITE TABLE page_view PARTITION(dt='2008-06-08', country='US')

      SELECT pvs.viewTime, … WHERE pvs.country = 'US'

  INSERT OVERWRITE TABLE page_view PARTITION(dt='2008-06-08', country='CA')

      SELECT pvs.viewTime, ... WHERE pvs.country = 'CA'

  INSERT OVERWRITE TABLE page_view PARTITION(dt='2008-06-08', country='UK')

      SELECT pvs.viewTime, ... WHERE pvs.country = 'UK';
```

Example: Joins

```
CREATE TABLE customer (id INT,name STRING,address STRING)
  ROW FORMAT DELIMITED FIELDS TERMINATED BY '#';

CREATE TABLE order_cust (id INT,cus_id INT,prod_id INT,price INT)
  ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';
```

- **SELECT** * **FROM** customer c **JOIN** order_cust o **ON** (c.id=o.cus_id);
- **SELECT** c.id, c.name, c.address, ce.exp

```
FROM customer c JOIN (SELECT cus_id,sum(price) AS exp
                      FROM order_cust
                      GROUP BY cus_id) ce ON (c.id=ce.cus_id);
```

Inserts into Files, Tables and Local Files

```
FROM pv_users
INSERT INTO TABLE pv_gender_sum
  SELECT pv_users.gender, count_distinct(pv_users.userid)
  GROUP BY(pv_users.gender)
INSERT INTO DIRECTORY '/user/tmp/dir'
  SELECT pv_users.age, count_distinct(pv_users.userid)
  GROUP BY(pv_users.age)
INSERT INTO LOCAL DIRECTORY '/home/local/dir'
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY \013
  SELECT pv_users.age, count_distinct(pv_users.userid)
  GROUP BY(pv_users.age);
```