



Data Processing Frameworks for Hadoop

HBD



Technology Map

“EXTRACT”
(Acquire)



“TRANSFORM”
(Process)



“LOAD”
(STORE)



APACHE
HBASE




Introduction

MapReduce was very useful to enable developers to **easily scale** their data processing capabilities.

Eventually the **need to democratize the access to Mapreduce's** processing power, speed up development, and generalize behaviour lead to frameworks that used **SQL or similar languages** to process data and this operations are automatically translated to MapReduce.

The first examples of this kind of frameworks that appeared where **Apache Hive** and **Apache Pig** which are still relevant today.





Pig

platform for analyzing large data sets that consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs.

Apache Pig Word count

```
input_lines = LOAD '/tmp/my-copy-of-all-pages-on-internet' AS(line: chararray);

--Extract words from each line then flatten the bag to get one word on each row
words = FOREACH input_lines GENERATE FLATTEN(TOKENIZE(line)) AS word;
--filter any words that are white spaces
filtered_words = FILTER words BY word MATCHES '\\w+';
--create a group for each word
word_groups = GROUP filtered_words BY word;
--count the entries in each group
word_count = FOREACH word_groups GENERATE COUNT(filtered_words) AS count, group AS word;
--order the records by count
ordered_word_count = ORDER word_count BY count DESC;

STORE ordered_word_count INTO '/tmp/number-of-words-on-internet';
```



Apache Pig

Pig is made up of two pieces:

- » **Pig Latin:** High level procedural language.
- » **Pig Engine:** Parser, Optimizer and distributed query execution.

Created at **Yahoo in 2008**. Used a lot at Netflix and Yahoo for ETLs.


Complex tasks comprised of **multiple interrelated data transformations are explicitly encoded as data flow sequences**, making them easy to write, understand, and maintain.

Pig can access Hive's tables definitions in the metastore through HCatalog.





Apache Pig Running modes

- » **Grunt Shell:** Enter Pig commands manually using Pig's interactive shell, Grunt.
 - » **Script File:** Place Pig commands in a script file and run the script.
 - » **Embedded Program:** Embed Pig commands in a host language and run the program.
 - » **Hue:** Use Hue's built in Pig query editor (comes with function assist).
 - » Eclipse, IntelliJ IDEA, Vim, Emacs, and TextMate have **plugins** that understand Pig Latin.
- 



Apache Pig Running shell

Local Mode:

To run Pig in local mode, you need access to a single machine:

```
$> pig -x local
```

Hadoop (MapReduce) Mode:

To run Pig in Hadoop (MapReduce) mode, you need access to a Hadoop cluster and HDFS installation.

```
$> pig [-x mapreduce]
```





Apache Pig Relational Operators

- » **LOAD:** loads data from file system.
- » **DUMP:** Displays results to screen.
- » **STORE:** Saves results to the file system.
- » **STREAM:** Sends data to an external script or program.
- » **FILTER:** Selects tuples from a relation based on some condition.
- » **JOIN:** Joins two or more relations.
- » **ORDER:** Sorts a relation.
- » **DISTINCT:** Removes duplicate tuples in a relation.
- » **FOREACH:** Makes a transformation.
- » **GROUP:** Groups data in a relation.
- » **COGROUP:** Same as Group.
- » **LIMIT:** Limits the amount of tuples.
- » **UNION:** Computes the union of 2 relations.
- » **SAMPLE:** Selects a ramdom set of tuples of the relation
- » **CROSS:** Computes the cross from 2 relation.
- » **SPLIT:** Partitions a relation into two or more relations.

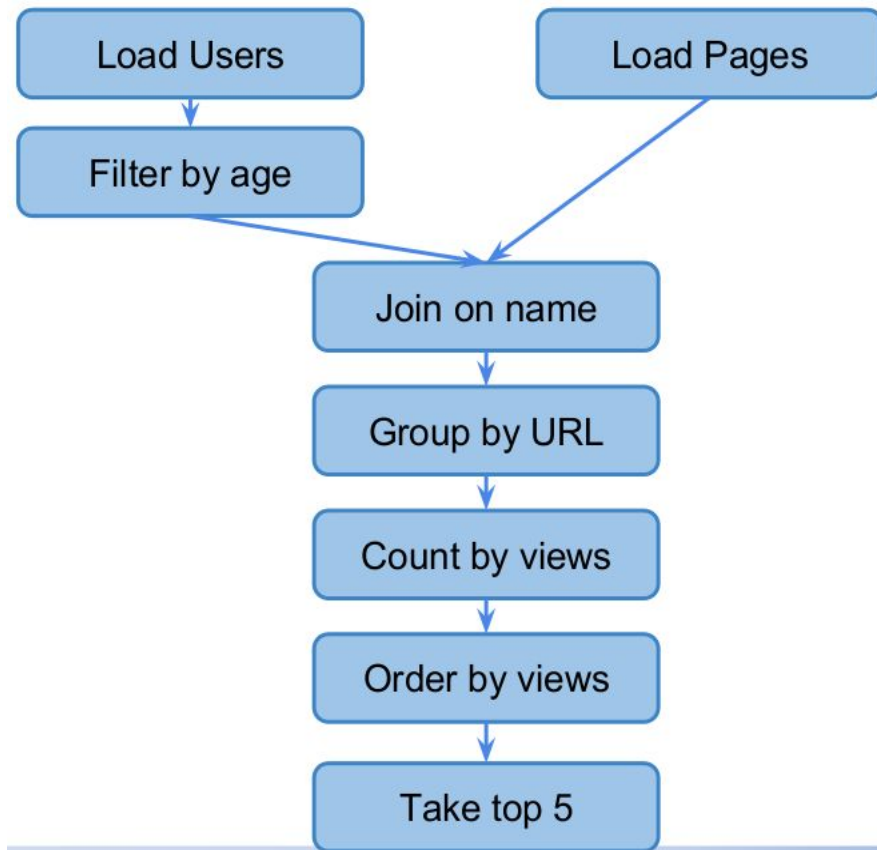
Apache Pig Example

Input:

- » User profiles.
- » Page visits.

Output:


- » The top 5 most visited pages by users aged 18-25






Apache Pig Example

```
users = LOAD 'users' AS (name:chararray, age:int);
users_filtered = FILTER users BY age >= 18 AND age <= 25;
pages = LOAD 'pages' AS (user:chararray, url:chararray);
joined = JOIN users_filtered BY name, pages BY user;
group_by_url = GROUP joined BY url;
clicks_count_by_url = FOREACH group_by_url GENERATE GROUP, COUNT(joined) AS clicks;
sorted_counts = ORDER clicks_count_by_url BY clicks DESC;
top_5_sites = LIMIT sorted_counts 5;
--DUMP top_5_sites;
--ILLUSTRATE top_5_sites;
--DUMP top_5_sites;
STORE top_5_sites INTO 'top5sites';
```





Apache Pig Aggregation Functions

- » **AVG**
 - » **CONCAT**
 - » **COUNT**
 - » **COUNT_STAR**
 - » **DIFF**
 - » **TOKENIZE**
 - » **IsEmpty**
 - » **MAX**
 - » **MIN**
 - » **SIZE**
 - » **SUM**
- 

Apache Pig User Defined Function

Java code:

```
package myudfs;

import java.io.IOException;
import org.apache.pig.EvalFunc;
import org.apache.pig.data.Tuple;

public class UPPER extends EvalFunc<String> {
    public String exec(Tuple input) throws IOException {
        if (input == null || input.size() == 0)
            return null;
        try{
            String str = (String)input.get(0);
            return str.toUpperCase();
        }catch(Exception e){
            throw new IOException("Caught exception processing input
row ", e);
        }
    }
}
```

```
REGISTER myudfs.jar;
```

```
A = LOAD 'student_data' AS (name:
chararray, age: int, gpa: float);
```

```
B = FOREACH A GENERATE myudfs.UPPER(name);
DUMP B;
```

[Pig UDF page](#)



Hive

Data warehouse infrastructure built on top of Hadoop for providing data summarization, query, and analysis.

Apache Hive: Word count

```
DROP TABLE IF EXISTS docs ;remove previous table
CREATE TABLE docs (line STRING); create staging table
LOAD data inpath 'input_file' overwrite INTO TABLE docs; load data into
staging table
; create word count table
CREATE TABLE word_counts AS
    SELECT    word, count(1) AS COUNT
    FROM      (SELECT Explode(Split(line, '\s')) AS word FROM docs) temp
    GROUP BY word
    ORDER BY word;
```



Apache Hive

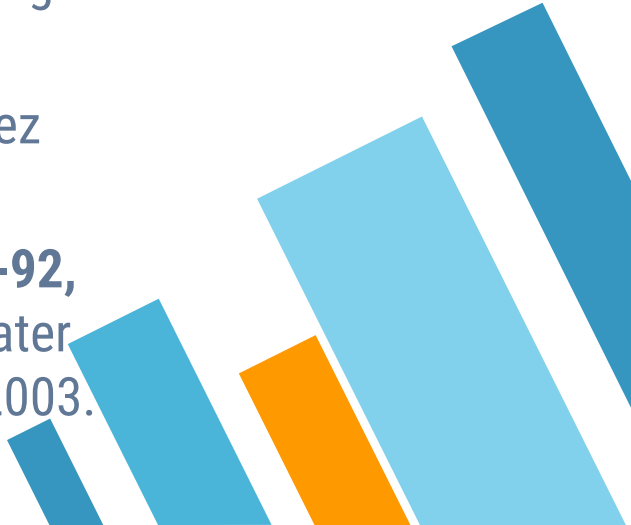
It was initially **developed by Facebook in 2008**. Later companies like Netflix and Hortonworks started contributing.

Oriented towards analytical (Business Intelligence)

workloads, not iterative workflows like machine learning algorithms.

In the beginning **used MapReduce as backend**, now Tez (DAG execution engine) and **Spark**

Hive's SQL dialect, called **HiveQL**, is a mixture of **SQL-92**, **MySQL**, **Oracle's SQL** dialect and also features from later SQL standards, such as window functions from SQL:2003.



Apache Hive Execution

Hive runs on the master machine

- » Converts HiveQL queries to MapReduce jobs.
- » Submits MapReduce jobs to the cluster.

HiveQL Statements

```
SELECT zipcode, SUM(cost) AS total
FROM customers JOIN orders
ON customers.id = orders.cid
WHERE zipcode LIKE '63%'
GROUP BY zipcode
ORDER BY total DESC;
```

Hive Interpreter / Execution Engine

- Parse HiveQL
- Make optimizations
- Plan execution
- Generate MapReduce jobs
- Submit job(s) to Hadoop
- Monitor progress



MapReduce Jobs



Apache Hive Storage

Hive queries operate on tables; just like any other RDBMS

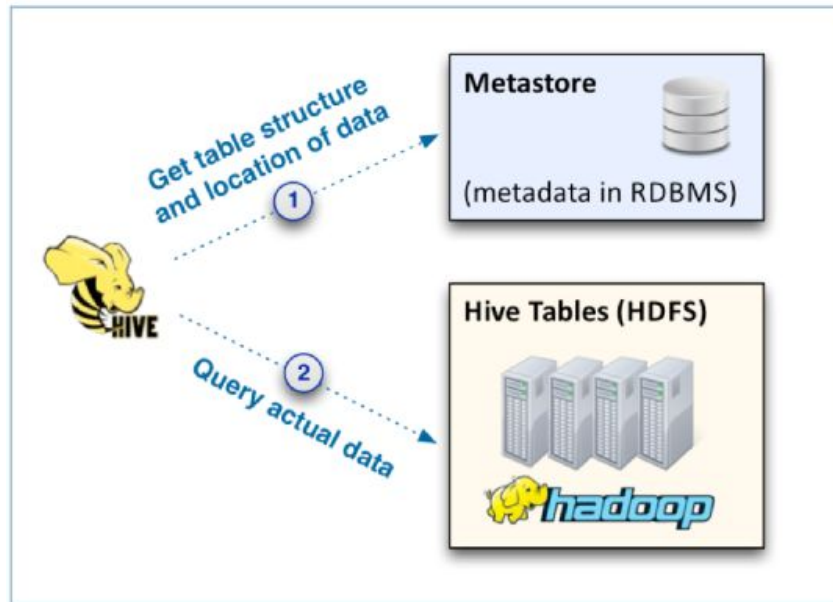
A table is just an HDFS directory containing one or more files

Default Path:

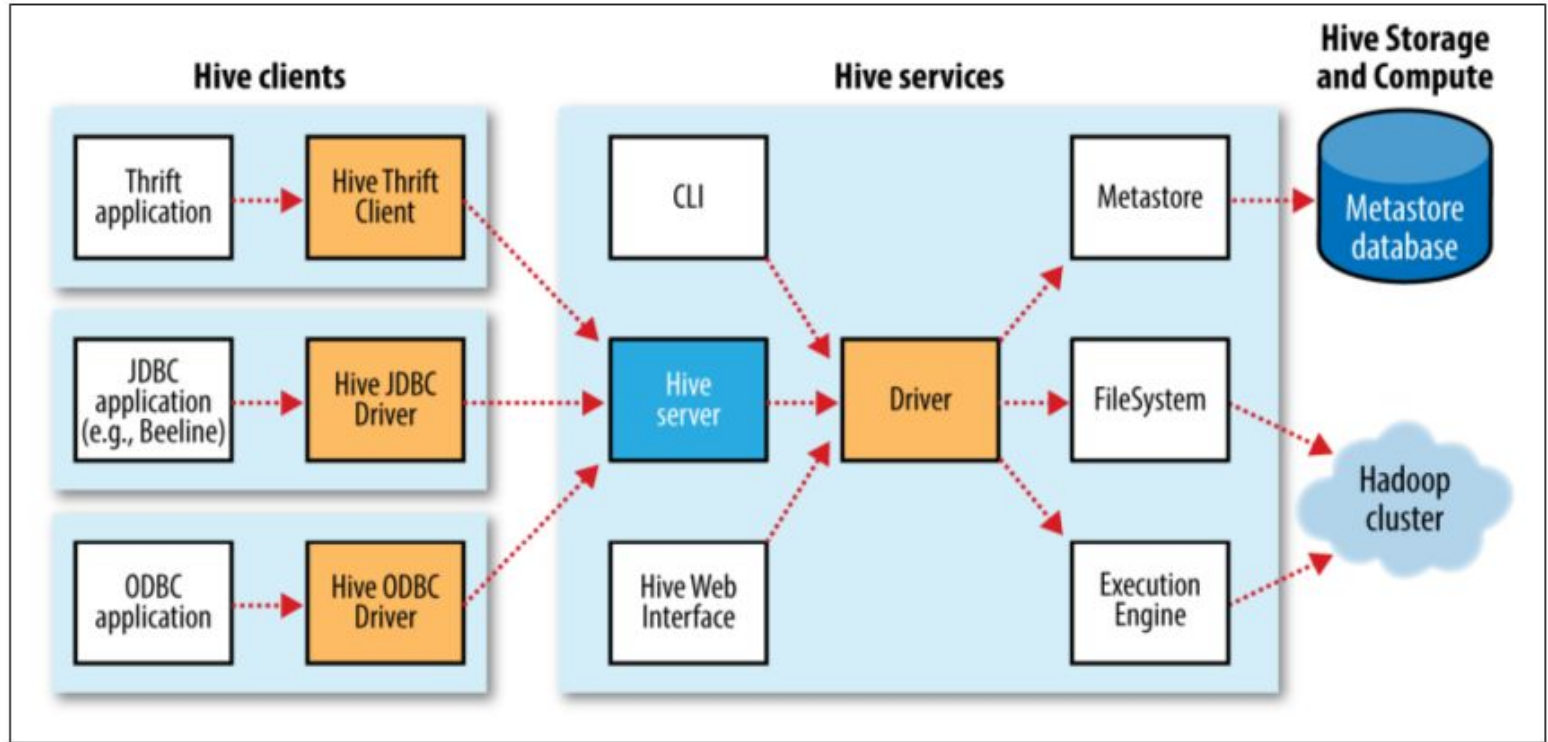
/user/hive/warehouse/<table_name>

Hive supports many formats for data stored

Hive stores the table metadata (location, fields, format, etc.) in the metastore a specialized process that uses an RDBMS to store data.




Apache Hive Architecture






Apache Hive Client Services

- » **cli**: shell command-line interface to Hive. Default service.
 - » **hiveserver2**: runs Hive as a server exposing a **Thrift service**. Supports authentication and multi-user concurrency. Applications using the Thrift, JDBC, and ODBC connectors need to run a Hive server to communicate with Hive.
 - » **beeline**: command-line interface to Hive that works in **embedded** mode (like the regular CLI), or by connecting to a **HiveServer 2** process using JDBC.
 - » **jar**: Hive equivalent of hadoop **jar** , a convenient way **to run Java applications** that includes both Hadoop and Hive classes on the classpath.
 - » **metastore**: by default, the metastore is run in the same process as the Hive service. Using this service, **it is possible to run the metastore as a standalone (remote) process**. Derby database by default, MySQL and PostgreSQL preferred.
- 



Apache Hive Shell Client

Located in `$HIVE_HOME/bin/hive`

- » **-e <quoted-query-string>** : SQL from command line.
 - » **-f <filename>**: SQL from files.
 - » **-h <hostname>**: Connecting to Hive Server on remote host.
 - » **--hiveconf <property=value>**: Use value for given property.
 - » **--hivevar <key=value>**: Variable substitution to apply to hive commands. e.g--hivevar A=B.
 - » **-i <filename>**: Initialization SQL file.
 - » **-p <port>**: Connecting to Hive Server on port number.
 - » **-S,--silent**: Silent mode in interactive shell.
 - » **-v,--verbose**: Verbose mode (echo executed SQL to the console).
 - » **--database <dbname>**: Specify the database to use.
- 

How to use it?

```
$> $HIVE_HOME/bin/hive options
```

```
$> hive -e 'select a.col from tab1 a' #running a query from the command line
```

```
$> hive -e 'select a.col from tab1 a' --hiveconf hive.exec.scratchdir=/home/my/hive_scratch  
--hiveconf mapred.reduce.tasks=32 #setting Hive configuration variables
```

```
$> hive -S -e 'select a.col from tab1 a' > a.txt #Example of dumping data out from a query into  
a file using silent mode
```

```
$> hive -f /home/my/hive-script.sql #Example of running a script non-interactively from local  
disk
```

```
$> hive -f hdfs://<namenode>:<port>/hive-script.sql #Example of running a script  
non-interactively from a Hadoop supported filesystem
```




Hive

Query Language



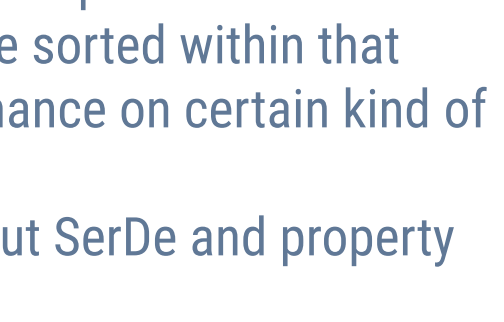


Apache Hive Create Table

- » **CREATE TABLE** creates a table with given name. An error is thrown if a table with the same name exists. You can use IF NOT EXISTS to skip the error.
 - » **EXTERNAL keyword** lets you create a table and keep the data after the table gets deleted
 - » You can provide a **LOCATION** so that Hive does not use default location for this table. This comes in handy if you already have data generated.
 - » You can create tables with **custom SerDe** or using native SerDe. A native SerDe is used if ROW FORMAT is not specified or ROW FORMAT DELIMITED is specified. You can use DELIMITED clause to read delimited files. **Use SERDE** clause to create a table with **custom SerDe**.
 - » You must specify list of columns for tables with native SerDe. Refer to Types part of the User Guide for the allowable column types. List of columns for tables with custom SerDe may be specified but Hive will query the SerDe to determine the list of columns for this table.
- 



Apache Hive Create Table

- » Use **STORED AS TEXTFILE** if the data needs to be stored as **plain text** files. Use **STORED AS SEQUENCEFILE** if the data needs to be compressed.
 - » **Partitioned tables** can be created using PARTITIONED BY clause. A table can have one or more partition columns and a separate data directory is created for each set of partition columns values. Further tables or partitions can be bucketed using CLUSTERED BY columns and data can be sorted within that bucket by SORT BY columns. This can improve performance on certain kind of queries.
 - » Table names and column names are case insensitive but SerDe and property names are case sensitive.
- 

Apache Hive Create table

```
CREATE TABLE page_view (  
    viewtime      INT,  
    userid        BIGINT,  
    page_url      STRING,  
    referrer_url  STRING,  
    ip            STRING comment 'IP Address of the User'  
)  
comment 'This is the page view table'  
partitioned BY(dt string, country string) stored AS sequencefile;
```

The statement above **creates page_view** table with viewTime, userid, page_url, referrer_url, up columns with a comment.

The table is also **partitioned** and data is stored in **sequence files**.

The data in the files assumed to be field delimited by ctrl-A and **row delimited by newline**.

Apache Hive Create table

```
CREATE TABLE page_view (  
    viewtime      INT,  
    userid        BIGINT,  
    page_url      STRING,  
    referrer_url  STRING,  
    ip            STRING comment 'IP Address of the User'  
)  
comment 'This is the page view table'  
partitioned BY(dt string, country string)  
clustered BY(userid) sorted BY(viewtime) INTO 32 buckets row format  
delimited fields TERMINATED BY '\001'  
    collection items TERMINATED BY '\002'  
    map KEYS TERMINATED BY '\003' LINES TERMINATED BY '\012'  
stored AS sequencefile;
```

The same table as previous table but the lines are delimited by '\012' instead of newline and bucketed(clustered by) userid and within each bucket the data is sorted in the increasing order of viewTime. Such an organization allows the user to do efficient sampling on the clustered column

Apache Hive Create table

```
CREATE EXTERNAL TABLE IF NOT EXISTS request_logs (  
    timestamp string,  
    event_type string,  
    request_ip string,  
    request_port int  
)  
partitioned BY (year string, month string, day string)  
row format delimited fields terminated BY '\t'  
                lines terminated BY '\n'  
row format serde 'org.apache.hadoop.hive.serde2.RegexSerDe'  
    WITH serdeproperties ( "input.regex" = "([^ ]*) ([^ ]*) ([^ ]*) : ([0-9]*) $" )  
stored AS [TEXTFILE|PARQUET|ORC|RCFILE|AVRO] location  
'[s3|hdfs]://logs.company.com/request_logs/';
```

Table **request_logs** partitioned by year, month, day with row limited by new line, and fields by tabs using **Regex serde** with **log format**.
Stored in a specific location.



Apache Hive Partitioned Tables

Partitions can be statically or dynamically added

```
ALTER TABLE request_logs ADDIF NOT EXISTS partition (year = '2015', month =  
'201505', day = '20150504') location  
's3://logs.company.com/request_logs/2015/05/04/';
```

```
ALTER TABLE request_logs ADDIF NOT EXISTS partition (year = '2015', month =  
'201505', day = '20150504') location  
's3://logs.company.com/request_logs/2015/05/05/';
```

If directories where in the format:

's3://logs.company.com/request_logs/year=2015/month=201505/day=20150504/'

We could use the following statement to load dynamically all partitions:

\$> MSCK REPAIR TABLE request_logs;





Apache Hive: Inserting Data from query

Staging table can be used to read data as it is and then use **insert-select** command to insert in the **final partitioned table**.

To use this Dynamic partitions must be enabled

```
set hive.exec.dynamic.partition=true;
set hive.exec.dynamic.partition.mode=nonstrict;
set hive.exec.max.dynamic.partitions.pernode=300;
set hive.exec.max.dynamic.partitions=9000;
set hive.exec.max.created.files=18000;
```

```
INSERT [OVERWRITE|INTO] TABLE
    db.requests_logs_parquet partition (
        year,
        month,
        day
    )
SELECT timestamp,
    event_type,
    request_ip,
    request_port, year,
    printf("%04d%02d", year, month) month,
    printf("%04d%02d%02d", year, month, day) day
FROM staging.requests_logs;
```



Apache Hive: Inserting Data from file

```
LOAD DATA [LOCAL] INPATH 'filepath' [OVERWRITE] INTO TABLE tablename [PARTITION  
(partcol1=val1, partcol2=val2 ...)]
```

filepath can be (both for a file or a directory):

- a relative path, such as project/data1
- an absolute path, such as /user/hive/project/data1
- a full URI with scheme `hdfs://namenode:9000/user/hive/project/data1`

The target being loaded to can be a table or a partition. If the table is partitioned, then one must specify a specific partition of the table by specifying values for all of the partitioning columns.

filepath can refer to a file (in which case Hive will move the file into the table) or it can be a directory (in which case Hive will move all the files within that directory into the table).

If the keyword **LOCAL** is specified, then Hive will look for the files in the local filesystem. if not then Hive will either use the full URI of filepath. In both cases Hive will move the **files addressed by filepath into the table** (or partition).

If the **OVERWRITE** keyword is used then the contents of the target table (or partition) will be deleted and replaced by the files referred to by filepath; otherwise the files referred by filepath will be added to the table.



Apache Hive: Managing Properties


Set properties priorities:

- » The Hive **SET** command.
- » The command-line **-hiveconf** option.
- » **hive-site.xml** and the **Hadoop site files** (core-site.xml, hdfs-site.xml, mapred-site.xml, and yarn-site.xml).
- » The Hive defaults and the Hadoop default files (core-default.xml, hdfs-default.xml, mapred-default.xml, and yarn-default.xml).

In the **Hive CLI** you can show all the currently set properties with the command:

```
$> set
```

A **configuration file** can be created that will be **executed** every time the **Hive CLI** starts and it has to be placed in **\$USER_HOME/.hiverc**.





Apache Hive: Querying Operands


Hive currently **supports** all common SQL-92 keywords like:

- » **SELECT**
 - » **FROM**
 - » **WHERE**
 - » **JOINS (LEFT, RIGHT, INNER, OUTER)**
 - » **UNION**
 - » **IN**
 - » **GROUP BY**
 - » **HAVING**
 - » **LIMIT**
 - » **EXISTS**
- 



Apache Hive: Special Querying Operands


Windowing options have been added

- » **LEAD:** The number of rows to lead can optionally be specified. If the number of rows to lead is not specified, the lead is one row. Returns null when the lead for the current row extends beyond the end of the window.
 - » **LAG:** The number of rows to lag can optionally be specified. If the number of rows to lag is not specified, the lag is one row. Returns null when the lag for the current row extends before the beginning of the window.
 - » **FIRST_VALUE**
 - » **LAST_VALUE**
- 



Apache Hive: Special Querying Operands

OVER clause with standard aggregates:

- » **COUNT**
 - » **SUM**
 - » **MIN**
 - » **MAX**
 - » **AVG**
 - » **OVER** with a **PARTITION BY** statement with one or more partitioning columns of any primitive datatype.
 - » **OVER** with **PARTITION BY** and **ORDER BY** with one or more partitioning and/or ordering columns of any datatype.
- 



Apache Hive: Special Querying Operands

Some additional analytical functions have been added like:

- » **RANK**
- » **ROW_NUMBER**
- » **DENSE_RANK**
- » **CUME_DIST**
- » **PERCENT_RANK**
- » **NTILE**

that can be used together with widow definitions with the OVER keyword.



Apache Hive Ejemplo Rank

```
hive> SELECT * FROM test4;
```

OK

USER_ID	VALUE	DESC
1	1	hallo
1	2	hallo
2	3	hallo1
2	7	hallo1
2	10	hallo3
2	11	hallo4

```
hive> SELECT *
```


```
FROM (SELECT user_id, value, desc,  
            Rank()  
            OVER (partition BY user_id  
                  ORDER BY value DESC) AS rank  
      FROM test4) t  
WHERE rank < 3;
```

OK

USER_ID	VALUE	DESC	RANK
1	2	hallo	1
1	1	hallo	2
2	11	hallo4	1
2	10	hallo3	2



Apache Hive: Ordering Operands

- » **DISTRIBUTED BY <field1, field2..>**: rows with the same values will be sent to the same reducer.
 - » **SORT BY <field1, field2..>**: rows in the same reducer will be sorted by the specified fields.
 - » **CLUSTER BY <field1, field2..>**: equal to combining DISTRIBUTED BY and SORTED BY over the same fields.
- 

Apache Hive: User Defined Functions

Java code:

```
package com.example.hive.udf;

import org.apache.hadoop.hive.q1.exec.UDF;
import org.apache.hadoop.io.Text;

public final class Lower extends UDF {
    public Text evaluate(final Text s) {
        if (s == null) { return null; }
        return new Text(s.toString().toLowerCase());
    }
}
```

Hive CLI:

```
hive> create temporary function my_lower
      as 'com.example.hive.udf.Lower';
hive> select my_lower(title),
      > sum(freq)
      > from titles
      > group by my_lower(title);
```



Apache Hive: Recent Improvements

Due to Hive's shortcomings (mostly originated by MapReduce) Cloudera announced it would be moving away from Hive and focus in Impala and Spark. Hortonworks announce the Stinger project and made the following improvements.

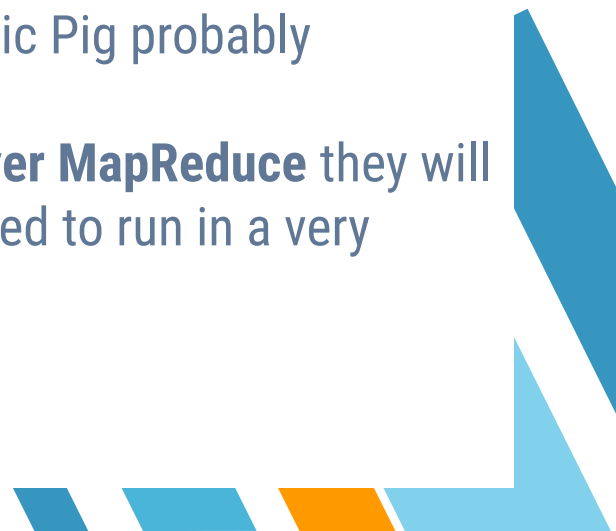
2.0	HBase metastore and a lot of Hive-on-Spark improvements.
1.0	HiveServer2 and improvements to Beeline.
0.14	Speed: cost-based optimizer for star and bushy join queries Scale: temporary tables Scale: transactions with ACID semantics (UPDATE, INSERT INTO, etc.)
0.13	Speed: Hive on Tez, vectorized query engine & cost-based optimizer Scale: dynamic partition loads and smaller hash tables SQL: CHAR & DECIMAL datatypes, subqueries for IN / NOT IN
0.12	Speed: Vectorized query engine & ORCFile predicate pushdown SQL: Support for VARCHAR and DATE semantics, GROUP BY on structs and unions

Pig vs. Hive

- » Pig Latin is **procedural**, where Hive SQL is **declarative**.
- » Pig Latin allows pipeline developers to decide where to **checkpoint data in the pipeline**.
- » Pig Latin allows the developer to select specific operator implementations directly rather than relying on the optimizer.
- » Pig Latin supports **splits in the pipeline**.
- » Pig Latin allows developers to insert their own code almost anywhere in the data pipeline.
- » **Hive** plays well with Business Intelligence and Data Visualization tools that use SQL to get their data.
- » **SQL** is most widely used and known.
- » **Flow control is difficult with Hive** and additional workflow tools are mostly needed for complex ETLs. **Pig, simplifies this a lot.**



Key Takeaways

- » Hive and Pig have been created a long time ago (in Hadoop terms).
 - » Both tools have constantly evolved and are now **moving to other execution backends**, specially pointing towards Spark.
 - » Both tools have their use case and their pros but **Hive has been more widely adopted** due to SQL's popularity.
 - » In **complex ETLs** with many steps and forks in the logic Pig probably presents a much clearer way of achieving this.
 - » Since these frameworks add a **layer of abstraction over MapReduce** they will probably not be the best choice for workloads that need to run in a very optimized way.
- 



CREDITS

Content of the slides:

- » Big Data Tools - ITBA

Images:

- » Big Data Tools - ITBA
- » obtained from: commons.wikimedia.org

Special thanks to all the people who made and released these awesome resources for free:

- » Presentation template by [SlidesCarnival](https://slidescarnival.com)
 - » Photographs by [Unsplash](https://unsplash.com)
- 