

Data **Cleansing** some important elements

Genoveva Vargas-Solar

CR1, CNRS, LIG-LAFMIA

Genoveva.Vargas@imag.fr

<http://vargas-solar.com>, Montevideo, 18nd November, 2014



HADAS
GROUP



Data cleansing/scrubbing

- Detecting and correcting (or removing) corrupt or inaccurate records from a record set, table, or database
- Used mainly in databases, the term refers to identifying incomplete, incorrect

Clean up the data in a database & **bring consistency** to different sets of data merged from separate databases

incomplete or missing entries are completed

- In more complex operations, data cleansing can be performed by computer programs that check the data with a variety of rules and procedures decided upon by the user

Dirty data

3

- Dummy Values
- Absence of Data
- Multipurpose Fields
- Cryptic Data
- Contradicting Data
- Inappropriate Use of Address Lines
- Violation of Business Rules
- Reused Primary Keys
- Non-Unique Identifiers
- Data Integration Problems

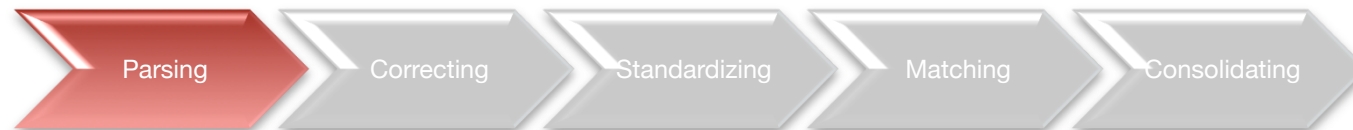
Data cleansing steps

Locates and identifies individual data elements in the source files and then isolates these data elements in the target files

Input Data from Source File
Beth Christine Parker, SLS MGR
Regional Port Authority
Federal Building
12800 Lake Calumet
Hedgewisch, IL



Parsed Data in Target File
First Name: Beth
Middle Name: Christine
Last Name: Parker
Title: SLS MGR
Firm: Regional Port Authority
Location: Federal Building
Number: 12800
Street: Lake Calumet
City: Hedgewisch
State: IL



Data cleansing steps

5

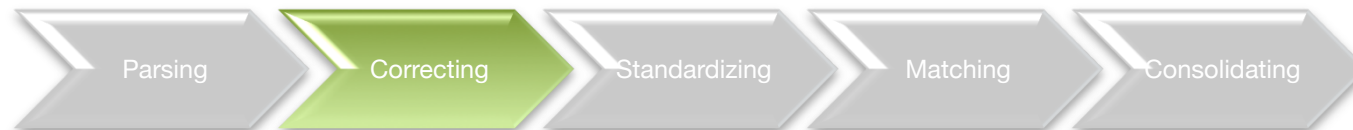
Corrects parsed individual data components using sophisticated data algorithms and secondary data sources

Parsed Data

First Name: Beth
Middle Name: Christine
Last Name: Parker
Title: SLS MGR
Firm: Regional Port Authority
Location: Federal Building
Number: 12800
Street: Lake Calumet
City: Hegewisch
State: IL

Corrected Data

First Name: Beth
Middle Name: Christine
Last Name: Parker
Title: SLS MGR
Firm: Regional Port Authority
Location: Federal Building
Number: 12800
Street: South Butler Drive
City: Chicago
State: IL
Zip: 60633
Zip+Four: 2398



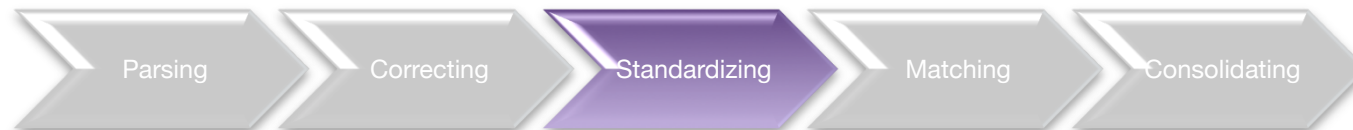
Data cleansing steps

Applies conversion routines to transform data into its preferred (and consistent) format using both standard and custom business rules.

Corrected Data
First Name: Beth
Middle Name: Christine
Last Name: Parker
Title: SLS MGR
Firm: Regional Port Authority
Location: Federal Building
Number: 12800
Street: South Butler Drive
City: Chicago
State: IL
Zip: 60633
Zip+Four: 2398



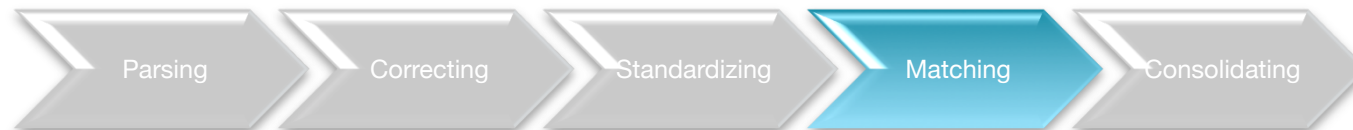
Corrected Data
Pre-name: Ms.
First Name: Beth
1st Name Match
Standards: Elizabeth, Bethany, Bethel
Middle Name: Christine
Last Name: Parker
Title: Sales Mgr.
Firm: Regional Port Authority
Location: Federal Building
Number: 12800
Street: S. Butler Dr.
City: Chicago
State: IL
Zip: 60633
Zip+Four: 2398



Data cleansing steps

Searching and matching records within and across the parsed, corrected and standardized data based on predefined business rules to eliminate duplicates

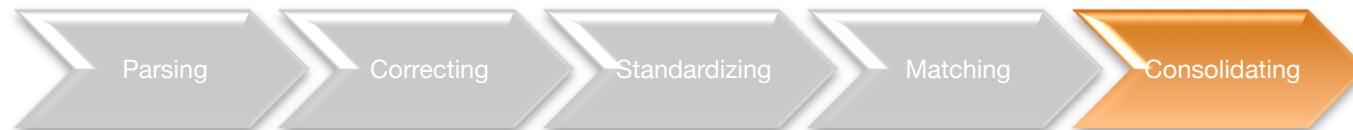
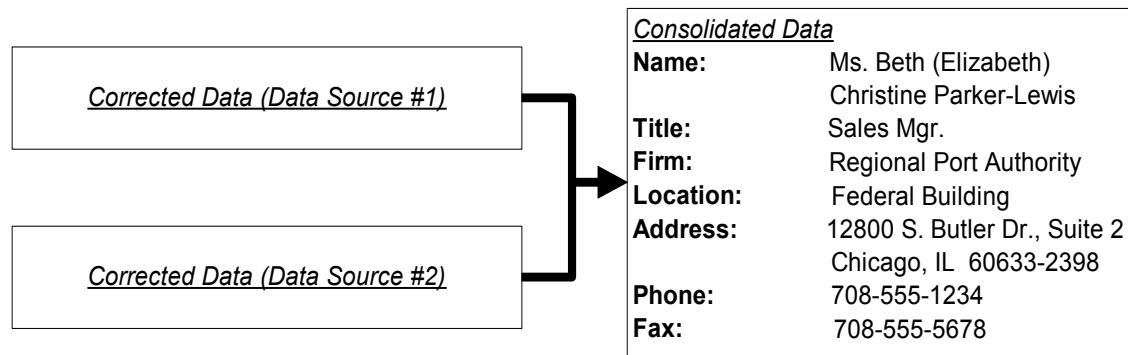
| <u>Corrected Data (Data Source #1)</u> | | <u>Corrected Data (Data Source #2)</u> | |
|--|----------------------------|--|-------------------------|
| Pre-name: | Ms. | Pre-name: | Ms. |
| First Name: | Beth | First Name: | Elizabeth |
| 1st Name Match | | 1st Name Match | |
| Standards: | Elizabeth, Bethany, Bethel | Standards: | Beth, Bethany, Bethel |
| Middle Name: | Christine | Middle Name: | Christine |
| Last Name: | Parker | Last Name: | Parker-Lewis |
| Title: | Sales Mgr. | Title: | |
| Firm: | Regional Port Authority | Firm: | Regional Port Authority |
| Location: | Federal Building | Location: | Federal Building |
| Number: | 12800 | Number: | 12800 |
| Street: | S. Butler Dr. | Street: | S. Butler Dr., Suite 2 |
| City: | Chicago | City: | Chicago |
| State: | IL | State: | IL |
| Zip: | 60633 | Zip: | 60633 |
| Zip+Four: | 2398 | Zip+Four: | 2398 |
| Phone: | | Phone: | 708-555-1234 |
| Fax: | | Fax: | 708-555-5678 |



Data cleansing steps

8

Analyzing and identifying relationships between matched records and consolidating/merging them into ONE representation.





Prashanth Babu

<http://twitter.com/P7h>

Data Sanitation with Pig

Genoveva Vargas-Solar

CR1, CNRS, LIG-LAFMIA

Genoveva.Vargas@imag.fr

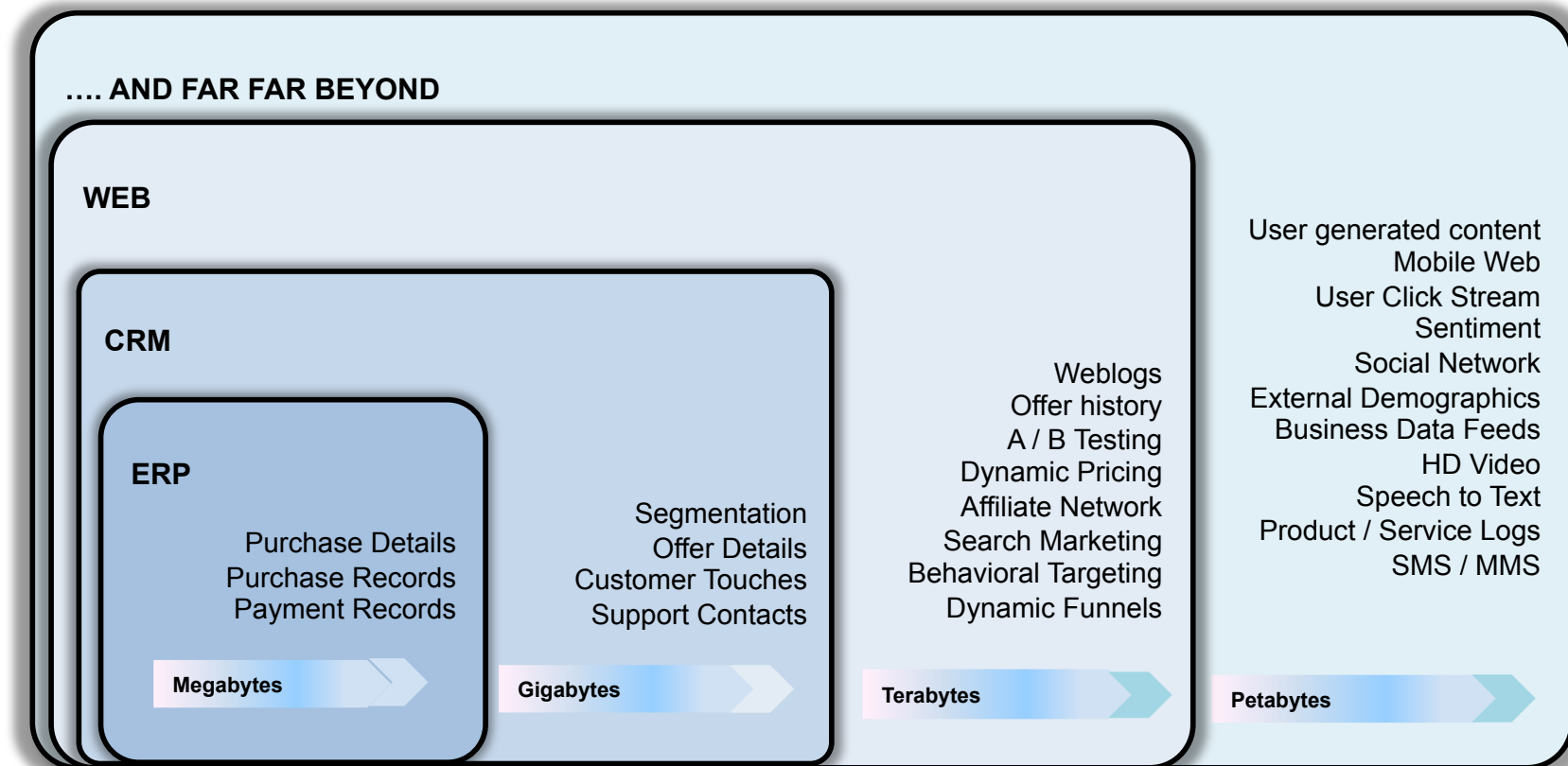
<http://vargas-solar.com>, Montevideo, 18nd November, 2014



HADAS
GROUP



Big data arena



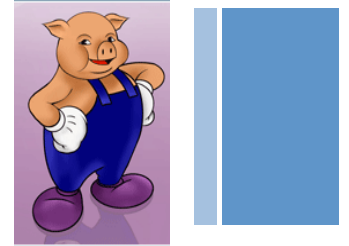
Source: <http://datameer.com>

Hadoop Ecosystem



Source: <http://indoos.wordpress.com/2010/08/16/hadoop-ecosystem-world-map/>

Pig



“Pig Latin: A Not-So-Foreign Language for Data Processing”

- Christopher Olston, Benjamin Reed, Utkarsh Srivastava, Ravi Kumar, Andrew Tomkins (Yahoo! Research)
- http://www.sigmod08.org/program_glance.shtml#sigmod_industrial_program
- <http://infolab.stanford.edu/~usriv/papers/pig-latin.pdf>

Pig

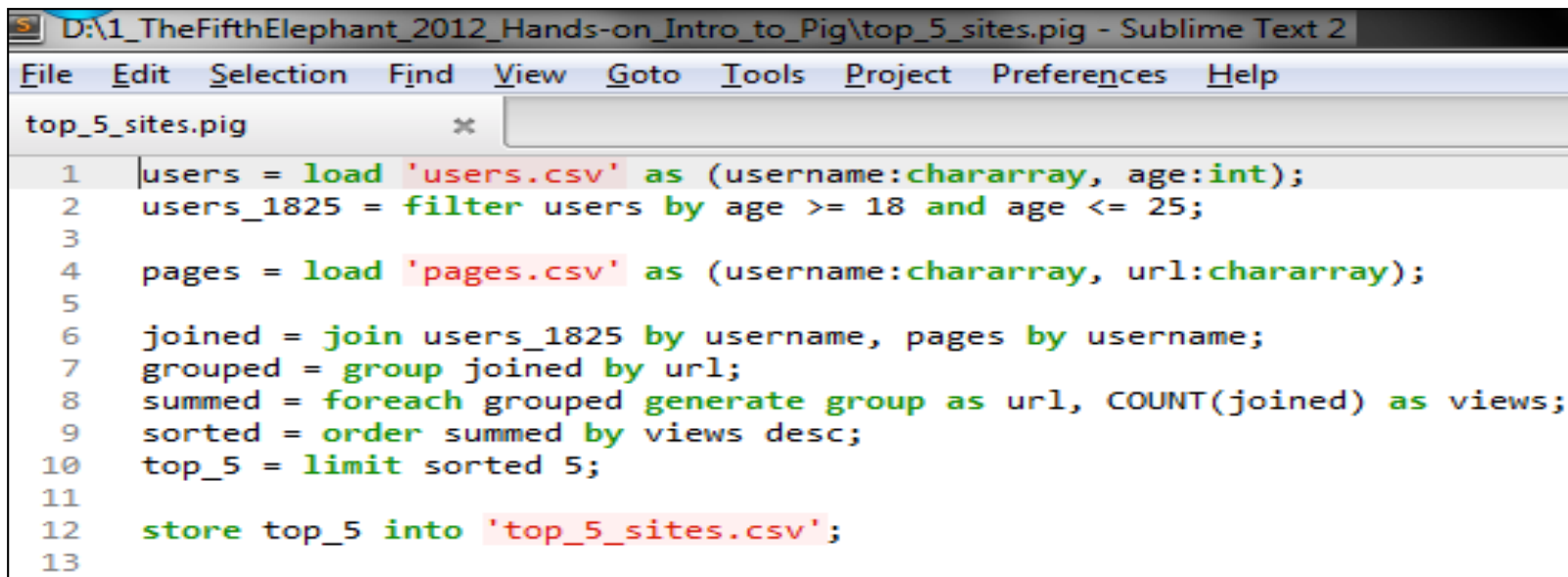
General description

- High level data flow language for exploring very large datasets
- Compiler that produces sequences of MapReduce programs
- Structure is amenable to substantial parallelization
- Operates on files in HDFS
- Metadata not required, but used when available
- Provides an engine for executing data flows in parallel on Hadoop

Key properties

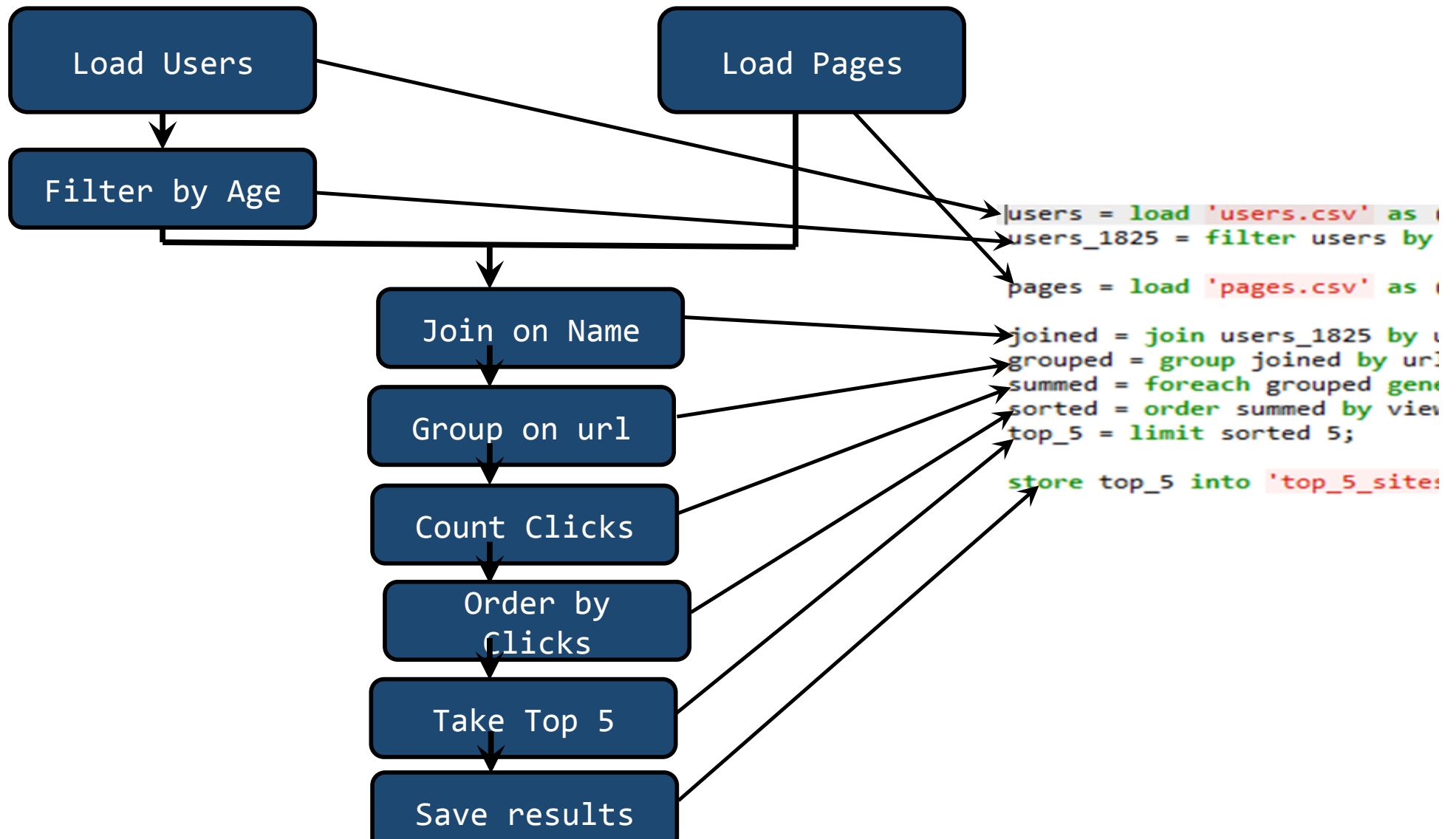
- **Ease of programming**
 - Trivial to achieve parallel execution of simple and parallel data analysis tasks
- **Optimization opportunities**
 - Allows the user to focus on semantics rather than efficiency
- **Extensibility**
 - Users can create their own functions to do special-purpose processing

Top 5 pages accessed by users between 18 and 25 year



The screenshot shows a Sublime Text editor window with the title bar "D:\1_TheFifthElephant_2012_Hands-on_Intro_to_Pig\top_5_sites.pig - Sublime Text 2". The menu bar includes File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, and Help. The editor displays a Pig script in the file "top_5_sites.pig". The script consists of 13 lines of code, with line numbers 1 through 13 on the left margin. The code is as follows:

```
1 users = load 'users.csv' as (username:chararray, age:int);
2 users_1825 = filter users by age >= 18 and age <= 25;
3
4 pages = load 'pages.csv' as (username:chararray, url:chararray);
5
6 joined = join users_1825 by username, pages by username;
7 grouped = group joined by url;
8 summed = foreach grouped generate group as url, COUNT(joined) as views;
9 sorted = order summed by views desc;
10 top_5 = limit sorted 5;
11
12 store top_5 into 'top_5_sites.csv';
13
```



Pig components

- **Pig Latin**

- Submit a script directly

- **Grunt**

- Pig Shell

- **PigServer**

- Java Class similar to JDBC interface

Execution modes

■ Local Mode

- Need access to a single machine
- All files are installed and run using your local host and file system
- Is invoked by using the -x local flag
- `pig -x local`

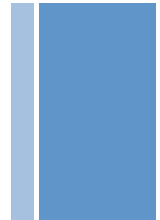
■ MapReduce Mode

- Mapreduce mode is the default mode
- Need access to a Hadoop cluster and HDFS installation
- Can also be invoked by using the -x mapreduce flag or just pig
 - `pig`
 - `pig -x mapreduce`

Statements

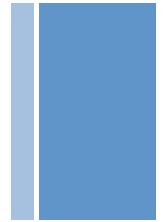
- Field is a piece of data
 - John
- Tuple is an ordered set of fields
 - (John, 18, 4.0F)
- Bag is a collection of tuples
 - (1, {(1, 2, 3)})
- Relation is a bag

Simple data types



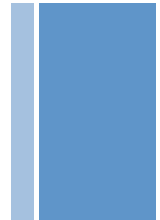
| Simple Type | Description | Example |
|-------------|--|---|
| int | Signed 32-bit integer | 10 |
| long | Signed 64-bit integer | Data: 10L or 10l Display: 10L |
| float | 32-bit floating point | Data: 10.5F or 10.5f or 10.5e2f or 10.5E2F Display: 10.5F or 1050.0F |
| double | 64-bit floating point | Data: 10.5 or 10.5e2 or 10.5E2 Display: 10.5 or 1050.0 |
| chararray | Character array (string) in Unicode UTF-8 format | hello world |
| bytearray | Byte array (blob) | |
| boolean | boolean | true/false (case insensitive) |

Complex data types



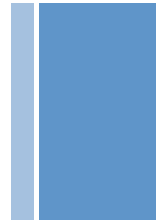
| Type | Description | Example |
|-------|---------------------------|------------------|
| tuple | An ordered set of fields. | (19,2) |
| bag | An collection of tuples. | {(19,2), (18,1)} |
| map | A set of key value pairs. | [open#apache] |

Commands



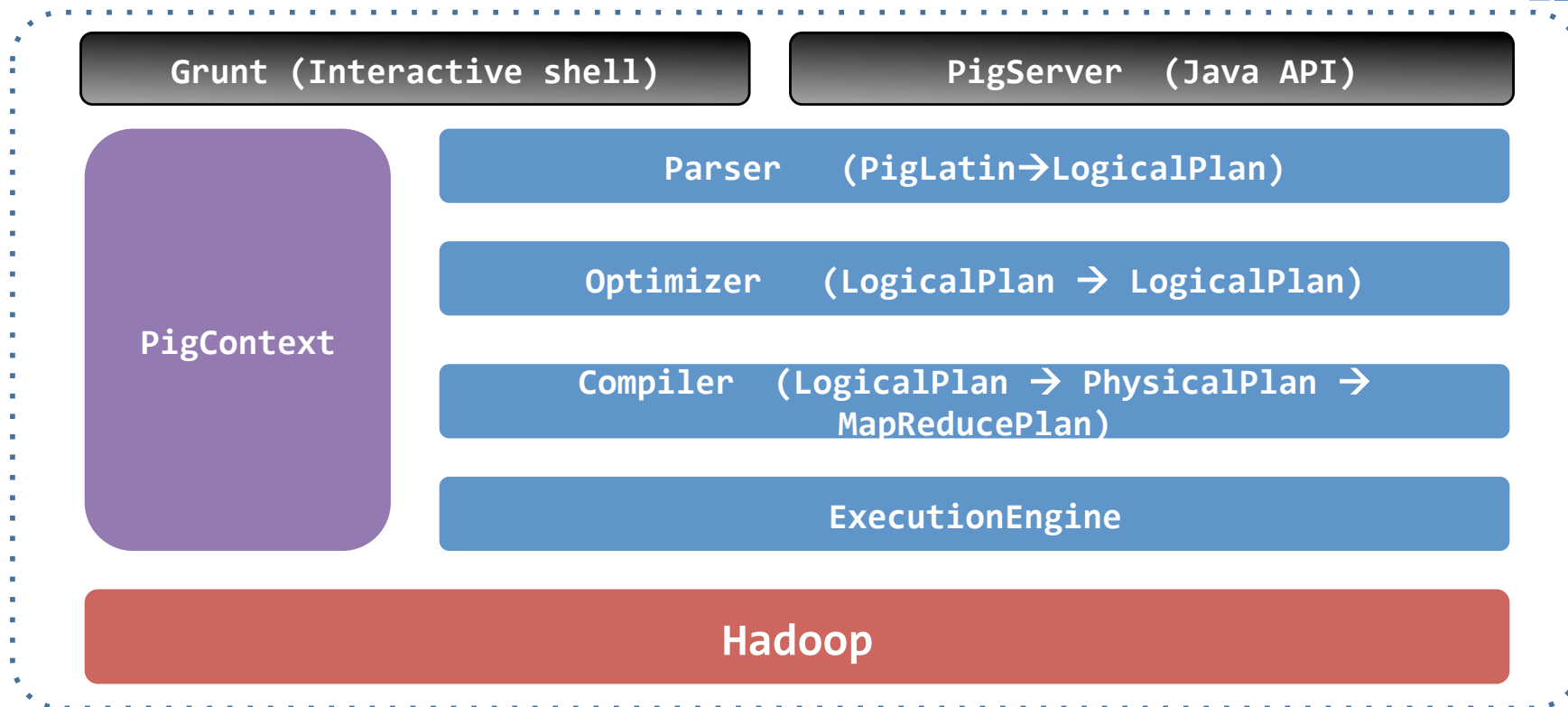
| Statement | Description |
|-----------------|--|
| Load | Read data from the file system |
| Store | Write data to the file system |
| Dump | Write output to stdout |
| Foreach | Apply expression to each record and generate one or more records |
| Filter | Apply predicate to each record and remove records where false |
| Group / Cogroup | Collect records with the same key from one or more inputs |
| Join | Join two or more inputs based on a key |
| Order | Sort records based on a Key |
| Distinct | Remove duplicate records |
| Union | Merge two datasets |
| Limit | Limit the number of records |
| Split | Split data into 2 or more sets, based on filter conditions |

Diagnostic operators



| Statement | Description |
|------------|---|
| Describe | Returns the schema of the relation |
| Dump | Dumps the results to the screen |
| Explain | Displays execution plans. |
| Illustrate | Displays a step-by-step execution of a sequence of statements |

Architecture



Pig Latin

```
countries = load '/user/gharriso/PIG_COUNTRIES' AS
(country_id, country_name , country_subregion , region);

customers= load '/user/gharriso/PIG_CUSTOMERS' AS
(cust_id,first_name, last_name, gender, yob, marital, postcode,city,country_id);

asianCountryys = filter countries by region matches 'Asia';

joined = join customers by country_id, asianCountryys by country_id;

grouped = group joined by country_name;

agged = foreach grouped generate group, COUNT(joined.customers::cust_id);

morethan500cust = filter agged by $1 > 500;

ordered =order morethan500cust by $1 desc;

dump ordered;
```

SQL or Hive QL

SELECT country_name,COUNT(cust_id) AS cust_count

FROM countries co

JOIN customers cu
ON (co.country_id=cu.country_id)

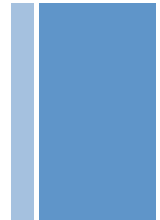
WHERE country_region='Asia'

GROUP BY country_name

HAVING COUNT(cust_id)>500

ORDER BY cust_count DESC

Pig vs. SQL



| Pig | SQL |
|------------------------------|--|
| Dataflow | Declarative |
| Nested relational data model | Flat relational data model |
| Optional Schema | Schema is required |
| Scan-centric workloads | OLTP + OLAP workloads |
| Limited query optimization | Significant opportunity for query optimization |

Source: <http://www.slideshare.net/hadoop/practical-problem-solving-with-apache-hadoop-pig>

Storage options with Pig

- HDFS
 - Plain Text
 - Binary format
 - Customized format (XML, JSON, Protobuf, Thrift, etc)
- RDBMS (DBStorage)
- Cassandra (CassandraStorage)
 - <http://cassandra.apache.org>
- HBase (HBaseStorage)
 - <http://hbase.apache.org>
- Avro (AvroStorage)
 - <http://avro.apache.org>

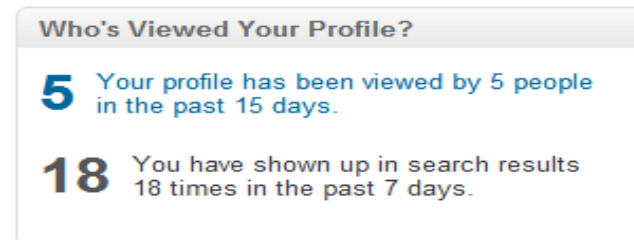
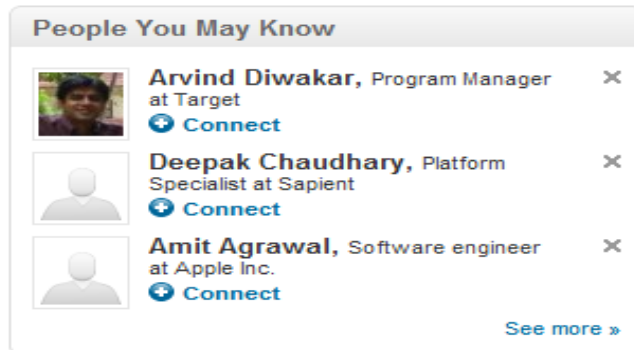
Using Pig

28

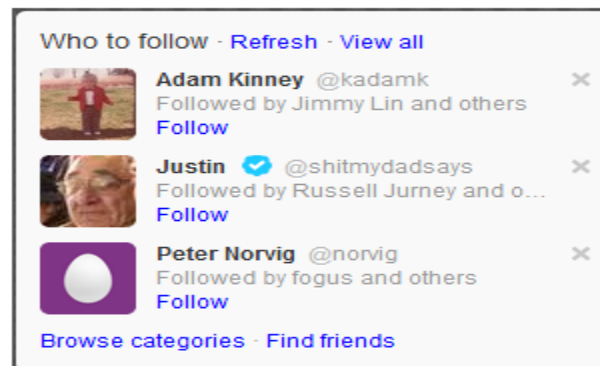
- Processing many Data Sources
- Data Analysis
- Text Processing
- Structured
- Semi-Structured
- ETL
- Machine Learning
- Advantage of sampling in any use case

Pig applications

LinkedIn



Twitter



Reporting, ETL, targeted emails & recommendations, spam analysis, ML

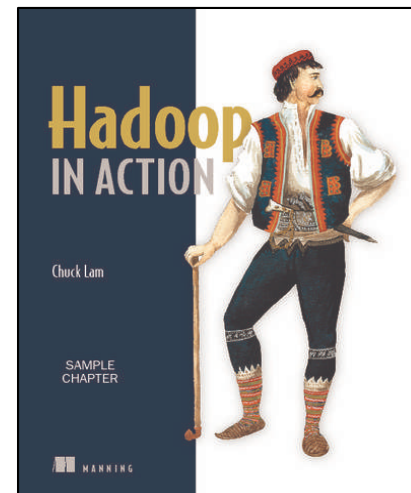


Source: <http://wiki.apache.org/pig/PoweredBy>

Books



<http://amzn.com/1449302645>



<http://amzn.com/1935182196>
Chapter:10 "**Programming with Pig**"

Useful references and links

- Online documentation: <http://pig.apache.org>
- Pig Confluence: <https://cwiki.apache.org/confluence/display/PIG/Index>
- Online Tutorials:
 - Cloudera Training, <http://www.cloudera.com/resource/introduction-to-apache-pig/>
 - Yahoo Training, <http://developer.yahoo.com/hadoop/tutorial/pigtutorial.html>
 - Using Pig on EC2: <http://developer.amazonwebservices.com/connect/entry.jspa?externalID=2728>
- Join the mailing lists:
 - Pig User Mailing list, user@pig.apache.org
 - Pig Developer Mailing list, dev@pig.apache.org
- Trainings and certifications
 - Cloudera: [http://university.cloudera.com/training/apache hive and pig/hive and pig.html](http://university.cloudera.com/training/apache%20hive%20and%20pig/hive%20and%20pig.html)
 - Hortonworks: <http://hortonworks.com/hadoop-training/hadoop-training-for-developers/>

