Apache Pig Latin

Rim Moussa

rim.moussa@gmail.com

Pig Latin: Building High-Level Dataflows over Map-Reduce

- High-level data processing language
  - Explicit data flow programming
- Pig's infrastructure layer consists of a compiler that produces sequences of Map-Reduce programs (workflow of MR jobs)
  - Manages all the details of connecting jobs and data flow
- Similar Projects
  - Sawzall (Google),
  - DryadLINK (MicroSoft)
- Execution modes
  - Local
  - Hadoop Cluster
- Piggybank.jar: open source UDFs

Pig Latin (ctnd.1)

- Dynamic schema, schema-free
- Data Model
  - A *relation* is a *bag* i.e. collection of tuples.
  - A *tuple* is an ordered set of *fields*.
  - A *field* is a piece of data.
- Data types:
  - int, long, float, double, chararray, bytearray, boolean, datatime, big integer, big decimal
  - Complex:
    - Tuple: ordered set of fields, e.g. (Sam, 20)
    - Bag: collection of tuples, e.g. {(Sam, 20), (Joe, 25)}
    - Map: a set of key-value pairs, e.g. [name#Sam, age#20]

Relational Operations

- Loading and Storing

LOAD	To Load the data from the file system (local/HDFS) into a relation.
STORE	To save a relation to the file system (local/HDFS)

- Filtering

FILTER BY	To remove unwanted rows from a relation.
DISTINCT	To remove duplicate rows from a relation.
FOREACH,GENERATE	To generate data transformations based on columns

- Grouping and Joining

JOIN	To join two or more relations.
COGROUP	To group the data in two or more relations.
GROUP	To group the data in a single relation.
CROSS	To create the cross product of two or more relations.

Relational Operations

- Sorting

SORT	To arrange a relation in a sorted order based on one or more fields (ascending or descending) on.
LIMIT	To get a limited number of tuples from a relation.

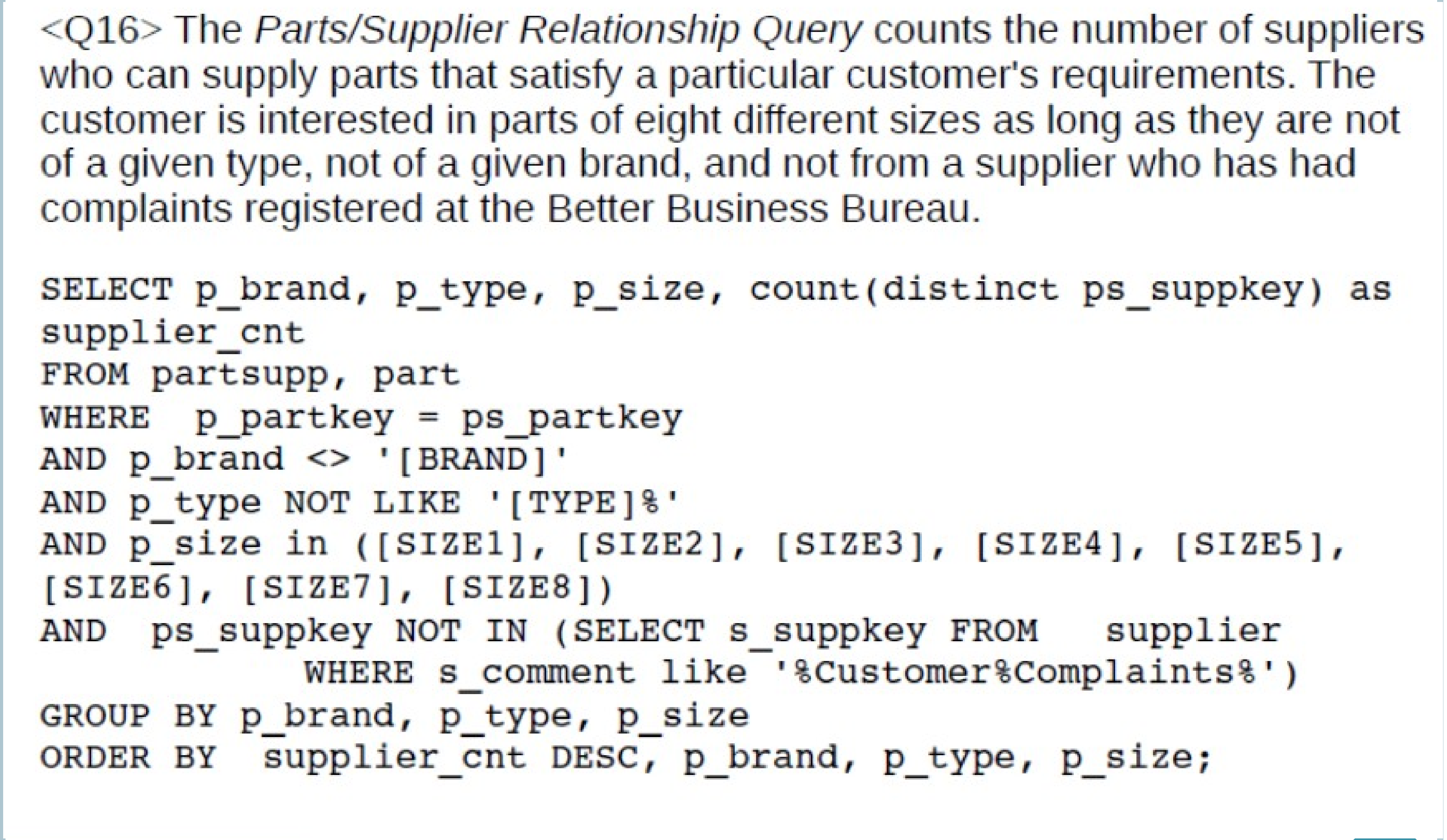
- Combining and Splitting

UNION	To combine two or more relations into a single relation.
SPLIT	To split a single relation into two or more relations.

- Diagnostic Operations

DUMP	To print the contents of a relation on the console
DESCRIBE	To describe the schema of a relation.
EXPLAIN	To view the logical, physical, or MapReduce execution plans
ILLUSTRATE	To view the step-by-step execution of a series of statements.

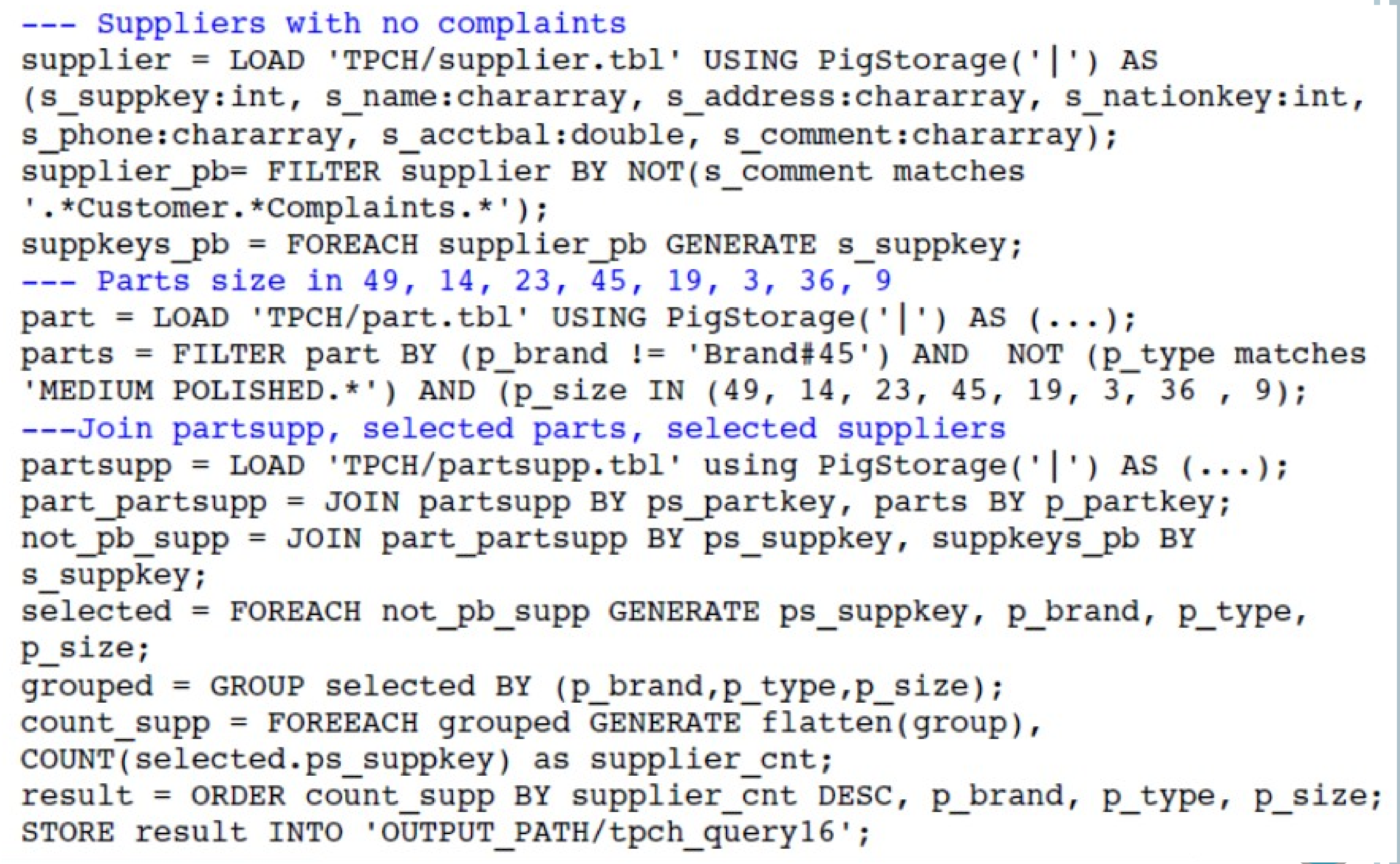
SQL statement of Q16 -TPC-H Benchmark



<Q16> The *Parts/Supplier Relationship Query* counts the number of suppliers who can supply parts that satisfy a particular customer's requirements. The customer is interested in parts of eight different sizes as long as they are not of a given type, not of a given brand, and not from a supplier who has had complaints registered at the Better Business Bureau.

```
SELECT p_brand, p_type, p_size, count(distinct ps_suppkey) as
supplier_cnt
FROM partsupp, part
WHERE  p_partkey = ps_partkey
AND p_brand <> '[BRAND]'
AND p_type NOT LIKE '[TYPE]%'
AND p_size in ([SIZE1], [SIZE2], [SIZE3], [SIZE4], [SIZE5],
[SIZE6], [SIZE7], [SIZE8])
AND  ps_suppkey NOT IN (SELECT s_suppkey FROM   supplier
            WHERE s_comment like '%Customer%Complaints%')
GROUP BY p_brand, p_type, p_size
ORDER BY  supplier_cnt DESC, p_brand, p_type, p_size;
```

Pig script of Q16 -TPC-H Benchmark



```
--- Suppliers with no complaints
supplier = LOAD 'TPCH/supplier.tbl' USING PigStorage('|') AS
(s_suppkey:int, s_name:chararray, s_address:chararray, s_nationkey:int,
s_phone:chararray, s_acctbal:double, s_comment:chararray);
supplier_pb= FILTER supplier BY NOT(s_comment matches
'.*Customer.*Complaints.*');
suppkeys_pb = FOREACH supplier_pb GENERATE s_suppkey;
--- Parts size in 49, 14, 23, 45, 19, 3, 36, 9
part = LOAD 'TPCH/part.tbl' USING PigStorage('|') AS (...);
parts = FILTER part BY (p_brand != 'Brand#45') AND  NOT (p_type matches
'MEDIUM POLISHED.*') AND (p_size IN (49, 14, 23, 45, 19, 3, 36 , 9);
---Join partsupp, selected parts, selected suppliers
partsupp = LOAD 'TPCH/partsupp.tbl' using PigStorage('|') AS (...);
part_partsupp = JOIN partsupp BY ps_partkey, parts BY p_partkey;
not_pb_supp = JOIN part_partsupp BY ps_suppkey, suppkeys_pb BY
s_suppkey;
selected = FOREACH not_pb_supp GENERATE ps_suppkey, p_brand, p_type,
p_size;
grouped = GROUP selected BY (p_brand,p_type,p_size);
count_supp = FOREEACH grouped GENERATE flatten(group),
COUNT(selected.ps_suppkey) as supplier_cnt;
result = ORDER count_supp BY supplier_cnt DESC, p_brand, p_type, p_size;
STORE result INTO 'OUTPUT_PATH/tpch_query16';
```

Relational Operations (ctnd. 2)

- Pig Supports different Join algorithms:
  - hash join --default,
  - merge join (using 'merge'),
  - skew (using 'skewed') ,
  - replication if small table (using 'replicated')
- Pig vs. MapReduce
  - Less code and less development effort
  - High level of abstraction
- Pig vs. SQL
  - *The step-by-step method of creating a program in Pig is much cleaner and simpler to use than the single block method of SQL. It is easier to keep track of what your variables are, and where you are in the process of analyzing your data*. (Jasmine Novak, eng. @ Yahoo!)

References

- C. Olston, B. Reed, U. Srivastava, R. Kumar and A. Tomkins. *Pig Latin: A Not-So-Foreign Language for Data Processing*. ACM SIGMOD 2008.
- Chuck Lam. *Hadoop in Action*. Manning. 2011.
- Tom White. *Hadoop: The Definitive Guide*. O'Reilly, Yahoo! Press. 2009. https://github.com/tomwhite/hadoop-book/
- R. Moussa, TPC-H Benchmarking of Apache Pig Latin in a Hadoop cluster /Grid5000, UTIC RR'2011.