Relational Algebra Operations and MapReduce

Relational Algebra Operators

- The relational algebra and the SQL language have many useful operators
 - Selection
 - Projection
 - Union, intersection, and difference
 - Join (see Join design patterns)
 - Aggregations and Group by (see the Summarization design patterns)

2

Relational Algebra Operators

- The MapReduce paradigm can be used to implement relational operators
 - However, the MapReduce implementation is efficient only when a full scan of the input table(s) is needed
 - i.e., when queries are not selective and process all data
 - Selective queries, which return few tuples/records of the input tables, are usually not efficient when implemented by using a MapReduce approach

Relational Algebra Operators

- Most preprocessing activities involve relational operators
 - E.g., the ETL processes in the data warehousing application context
 - E.g., the computation of the friends of a user

4

Relations/Tables

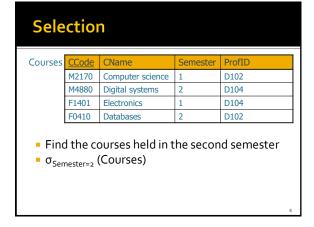
- Relations/Tables (also the big ones) can be stored in the HDFS distributed file system
 - They are broken in blocks and spread across the servers of the Hadoop cluster

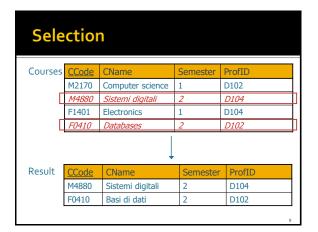
Relations/Tables

- Note
 - In relational algebra, relations/tables do not contain duplicate records by definition
 - This constraint must be satisfied by both the input and the output relations/tables

Selection

- $\sigma_{c}(R)$
 - Apply predicate (condition) C to each record of table R
 - Produce a relation containing only the records that satisfy predicate C
- The selection operator can be implemented by using the filtering pattern





Map-only job Mappers Analyze one record at a time of its split If the record satisfies C then emit a (key,value) pair with key=record and value=null Otherwise discard the record

Projection

- $-\pi_s(R)$
 - For each record of table R, keep only the attributes in S
 - Produce a relation with a schema equal to S (i.e., a relation containing only the attributes in S)
 - Remove duplicates, if any

Professors

Professors

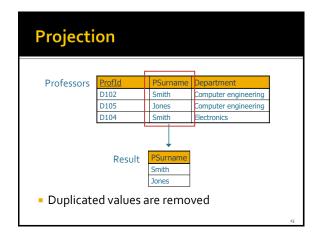
Professors

Professors

Profid
D102
Smith
Computer engineering
D105
Jones
Computer engineering
D104
Smith
Electronics

Find the surnames of all professors

π_{PSurname}(Professors)



Projection

- Mappers
 - Analyze one record at a time of its split
 - For each record r in R, select the values of the attributes in S and construct a new record r'
 - Emit a (key,value) pair with key=r' and value=null
- Reducers
 - Emit one (key, value) pair for each input (key, [list of values]) pair with key=r' and value=null

14

Union

- $-R \cup S$
 - R and S have the same schema
 - Produce a relation with the same schema of R and S
 - There is a record t in the output of the union operator for each record t appearing in R or S
 - Duplicated records are removed

Union

DegreeCourseProf

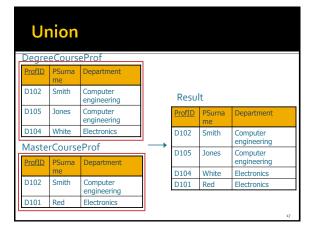
<u>ProfID</u>	PSurname	Department	
D102	Smith	Computer engineering	
D105	Jones	Computer engineering	
D104	White	Electronics	

MasterCourseProf

<u>ProfID</u>	PSurname	Department	
D102	Smith	Computer engineering	
D101	Red	Electronics	

- Find information relative to the professors of degree courses or master's degrees
- DegreeCourseProf ∪ MasterCourseProf

16



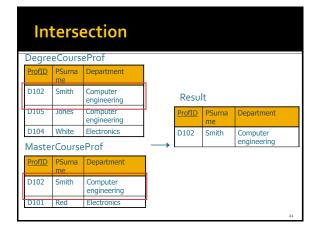
Union

- Mappers
 - For each input record t in R, emit one (key, value) pair with key=t and value=null
 - For each input record t in S, emit one (key, value) pair with key=t and value=null
- Reducers
 - Emit one (key, value) pair for each input (key, [list of values]) pair with key=t and value=null
 - i.e., one single copy of each input record is emitted

Intersection

- $\blacksquare R \cap S$
 - R and S have the same schema
 - Produce a relation with the same schema of R and S
 - There is a record t in the output of the intersection operator if and only if t appears in both relations (R and S)

Intersection								
	DegreeCourseProf							
	ProfID	PSurname	Department					
	D102	Smith	Computer engineering					
	D105	Jones	Computer engineering					
	D104	White	Electronics					
	MasterCourseProf							
	<u>ProfID</u>	PSurname	Department					
	D102	Smith	Computer engineering					
	D101	Red	Electronics					
 Find information relative to professors teaching both degree courses and master's courses DegreeCourseProf ∩ MasterCourseProf 								



Intersection

- Mappers
 - For each input record t in R, emit one (key, value) pair with key=t and value=t
 - For each input record t in S, emit one (key, value) pair with key=t and value=t

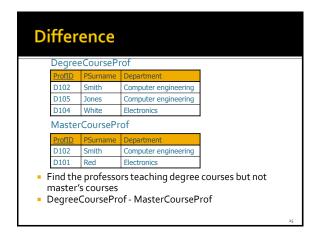
22

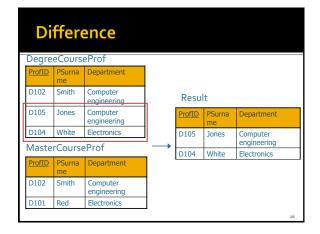
Intersection

- Reducers
 - Emit one (key, value) pair with key=t and value=null for each input (key, [list of values]) pair with [list of values] containing two values
 - It happens if and only if both R and S contain t

Difference

- R S
 - R and S have the same schema
 - Produce a relation with the same schema of R and S
 - There is a record t in the output of the difference operator if and only if t appears in R but not in S





Difference

- Mappers
 - For each input record t in R, emit one (key, value) pair with key=t and value=name of the relation (i.e., R)
 - For each input record t in R, emit one (key, value) pair with key=t and value=name of the relation (i.e., S)
- Two mapper classes are needed
 - One for each relation

Difference

- Reducers
 - Emit one (key, value) pair with key=t and value=null for each input (key, [list of values]) pair with [list of values] containing only the value R
 - It happens if and only if t appears in R but not in S

.

Join

- The join operators can be implemented by using the Join pattern
 - By using the reduce side or the map side pattern depending on the size of the input relations/tables

Aggregations and Group by

 Aggregations and Group by are implemented by using the Summarization pattern