Relational Algebra Operations and MapReduce

Table of contents

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)

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.

Most preprocessing activities involve relational operators (e.g., ETL processes in the data warehousing application context).

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

Notice that in relational algebra, relations/tables do not contain duplicate records by definition, and this constraint must be satisfied by both the input and the output relations/tables.

1 Selection

$$\sigma_C(R)$$

Selection applies predicate (condition) C to each record of table R, and produces a relation containing only the records that satisfy predicate C.

The selection operator can be implemented by using the filtering pattern.

ℹ Example

Given the table *Courses*

CCode	CName	Semester	ProfID
M2170	Computer science	1	D102
M4880	Digital systems	2	D104
F1401	Electronics	1	D104
F0410	Databases	2	D102

Find the courses held in the second semester

$$\sigma_{\textbf{Semester}=2}(\textbf{Courses})$$

The resulting table is

CCode	CName	Semester	ProfID
M4880	Digital systems	2	D104
F0410	Databases	2	D102

Selection is a map-only job, where each mapper analyzes one record at a time of its split and, if the record satisfies C then it emits a `(key,value)` pair with `key=record` and `value=null`, otherwise, it discards the record.

2 Projection

$$\pi_S(R)$$

Projection, for each record of table R, keeps only the attributes in S. It produces a relation with a schema equal to S (i.e., a relation containing only the attributes in S), an it removes duplicates, if any.

ℹ Example

Given the table *Professors*

ProfId	PSurname	Department
D102	Smith	Computer Engineering
D105	Jones	Computer Engineering
D104	Smith	Electronics

Find the surnames of all professors.

$$\pi_{\textbf{PSurname}}(\textbf{Professors})$$

The resulting table is

PSurname
Smith
Jones

Notice that duplicated values are removed.

In a projection

- Each mapper analyzes one record at a time of its split, and, for each record r in R, it selects the values of the attributes in S and constructs a new record r', and emits a `(key,value)` pair with `key=r'` and `value=null`.
- Each reducer emits one `(key, value)` pair for each input `(key, [list of values])` pair with `key=r'` and `value=null`.

3 Union

$$R \cup S$$

Given that R and S have the same schema, an union produces 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.

ℹ Example

Given the tables *DegreeCourseProf*

ProfId	PSurname	Department
D102	Smith	Computer Engineering
D105	Jones	Computer Engineering
D104	White	Electronics

and *MasterCourseProf*

ProfId	PSurname	Department
D102	Smith	Computer Engineering
D101	Red	Electronics

Find information relative to the professors of degree courses or master's degrees.

$$\textbf{DegreeCourseProf} \cup \textbf{MasterCourseProf}$$

The resulting table is

ProfId	PSurname	Department
D102	Smith	Computer Engineering
D105	Jones	Computer Engineering
D104	White	Electronics
D101	Red	Electronics

In a union

- Mappers, for each input record t in R, emit one `(key, value)` pair with `key=t` and `value=null`, and 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).

4 Intersection

$$R \cap S$$

Given that R and S have the same schema, an intersection produces 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).

ℹ Example

Given the tables *DegreeCourseProf*

ProfId	PSurname	Department
D102	Smith	Computer Engineering
D105	Jones	Computer Engineering
D104	White	Electronics

and *MasterCourseProf*

ProfId	PSurname	Department
D102	Smith	Computer Engineering
D101	Red	Electronics

Find information relative to professors teaching both degree courses and master's courses.

$$\mathbf{DegreeCourseProf} \cap \mathbf{MasterCourseProf}$$

The resulting table is

ProfId	PSurname	Department
D102	Smith	Computer Engineering

In an intersection

- Mappers, for each input record t in R, emit one `(key, value)` pair with `key=t` and `value="R"`, and For each input record t in S, emit one `(key, value)` pair with `key=t` and `value="S"`.
- 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. Notice that it happens if and only if both R and S contain t.

5 Difference

$$R - S$$

Given that R and S have the same schema, a difference produces 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.

ℹ Example

Given the tables *DegreeCourseProf*

ProfId	PSurname	Department
D102	Smith	Computer Engineering
D105	Jones	Computer Engineering
D104	White	Electronics

and *MasterCourseProf*

ProfId	PSurname	Department
D102	Smith	Computer Engineering
D101	Red	Electronics

Find the professors teaching degree courses but not master's courses.

$$\textbf{DegreeCourseProf} - \textbf{MasterCourseProf}$$

The resulting table is

ProfId	PSurname	Department
D105	Jones	Computer Engineering
D104	White	Electronics

In a 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 S, emit one `(key, value)` pair with `key=t` and `value=name` of the relation (i.e., S). Notice that two mapper classes are needed: one for each relation.
- 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. Notice that it happens if and only if t appears in R but not in S.

6 Join

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

7 Aggregations and Group by

Aggregations and Group by are implemented by using the Summarization pattern.