Algorytmy skalowalnego przetwarzania danych — projekt 2

dr Piotr Przymus, mgr Mikołaj Fejzer, dr Krzysztof Rykaczewski

14 kwietnia 2020

Spis treści

1 Algebra relacyjna

Relacja to tabela z kolumnami nazywanymi *atrybutami*. Zbiór atrybutów to *schemat relacji*. Element (wiersz) relacji nazywany jest *tuple*. Relację `R` o atrybutach `A` będziemy oznaczać `R(A)`.

Duże relacje można trzymać w wielu plikach na kilku maszynach. Wiele zapytań w bazach danych można rozbić na operacje podstawowe, dlatego są one wdzięcznym tematem dla MapReduce.

Na relacjach o tym samym schemacie można wykonywać operacje: sumy, przecięcia, różnicy. Połączenie dwóch relacji (*join*) można wykonywać dla dowolnych dwóch relacji. Połączenie powstaje w ten sposób, że dla każdych dwóch wierszy, jeśli zgadzają się one na wspólnym zbiorze atrybutów relacji, to łączymy te wiersze w jeden.

Uwaga: przez relację tutaj rozumiemy zbiór, a nie bag, tzn. każdy wiersz występuje tylko raz.

2 Zadanie: Operacje relacyjne

Napisać funkcje Map i Reduce, które pozwolą policzyć poniższe podpunkty.

1. Unię (SQL UNION) $A \cup B$,
2. Przecięcie (SQL INTERSECT) $A \cap B$,
3. Różnicę (SQL EXCEPT) $A \setminus B$.
4. INNER JOIN `Customers JOIN Orders`.
5. Selekcję `SELECT * FROM A WHERE condition;`.
6. Rzutowanie (*projection*) `SELECT x, y, z FROM A`.
7. Grupowanie i agregacja (*grouping and aggregation*).
8. (LEFT|RIGHT) OUTER JOIN

Przykładowe dane:

- punkty 1-3:
  - `relations_small.txt` – plik zawiera dwie kolumny, z których pierwsza to nazwa zbioru, a druga to element zbioru.
  - `relations_big.txt` – plik zawiera dwie kolumny, z których pierwsza to nazwa zbioru, a druga to element zbioru.
- punkty 1 i 8:
  - `join_data.txt` – plik zawiera dwie relacje `Customers` oraz `Orders`, które można złączyć.
- punkty 5-7:
  - `person.csv` – plik zawiera wiele kolumn pozwalających na selekcję, rzutowanie, grupowanie i agregację.

2.1 Unia (*union*) - 10 pkt

```
SELECT * FROM A
UNION
SELECT * FROM B;
```

2.1.1 Funkcja `map`

```
map(t) -> (t, t)
```

Przez `t` oznaczyliśmy *tuple* (wiersz relacji).

2.1.2 Funkcja `reduce`

```
reduce(t, [t, t, ..., t]) -> (t, t)
```

2.2 Przecięcie (*intersection*) - 10 pkt

```
SELECT * FROM A
INTERSECT
SELECT * FROM B;
```

2.2.1 Funkcja `map`

```
map(t) -> (t, t)
```

2.2.2 Funkcja `reduce`

Funkcja `reduce` działa w ten sposób, że jeśli dla jakiegoś klucza po posortowaniu mamy wartość `[t, t]`, to wypisujemy `(t, t)`, a w przeciwnym przypadku nic nie wypisujemy.

2.3 Różnica (*difference*) - 10 pkt

```
SELECT * FROM A
EXCEPT
SELECT * FROM B;
```

2.3.1 Funkcja `map`

`haskel` `map(t) -> (t, R)` gdzie `R` to odpowiednia relacja `A` lub `B`.

2.3.2 Funkcja `reduce`

Jeśli dla danego klucza po posortowaniu mamy wartość `[R]`, to wypisz `(t, t)`, a w przeciwnym przypadku nic nie wypisuj.

2.4 Połączenie (*inner join*) - 15 pkt

```
SELECT * FROM R, S WHERE R.B = S.B;
```

Załóżmy, że mamy relacje `R(A, B)` oraz `S(B, C)`, tzn. takie, które mają wspólny zbiór atrybutów `B`.

2.4.1 Funkcja `map`

```
map(a, b) -> (b, (R, a))
map(b, c) -> (b, (S, c))
```

2.4.2 Funkcja `reduce`

```
reduce(b, [(R, a), (S, c)]) -> (b, (a, b, c))
```

2.5 Selekcja (*selection*) - 10 pkt

```
SELECT * FROM A WHERE condition;
```

2.5.1 Funkcja `map`

Dla każdego tuple sprawdź, czy `t` spełnia warunek z selekcji i wtedy wypisz `(t, t)`.

2.5.2 Funkcja `reduce`

Funkcja `reduce` to identyczność.

2.6 Rzutowanie (*projection*) - 10 pkt

Dla każdego wiersza wypisz tylko wybrane kolumny. W SQL to

```
SELECT x, y, z FROM A;
```

2.6.1 Funkcja `map`

```
map(t) -> (t, t')
```

gdzie `t'` powstaje z wiersza `t` poprzez uwzględnienie tylko tych kolumn, które są zawarte w rzutowaniu.

2.6.2 Funkcja `reduce`

```
reduce(t, [t', t', ..., t']) -> (t, t')
```

2.7 Grupowanie i agregacja (*grouping and aggregation*) - 15 pkt

```
SELECT a, function(b) FROM A;
```

Załóżmy, że mamy relację `R(A, B, C)` i chcemy grupować po atrybutach `A`, a agregować po `B` (`C` są pomijane).

2.7.1 Funkcja `map`

```
map(a, b, c) -> (a, b)
```

2.7.2 Funkcja `reduce`

```
reduce(a, [b1, b2, ..., bn]) -> (a, function(b1, b2, ..., bn))
```

2.8 (LEFT|RIGHT) OUTER JOIN - 20 pkt

Bazując na wcześniejszym kodzie przygotuj przykłady (LEFT|RIGHT) OUTER JOIN.