BIG DATA - REPORT I PROGETTO

I JOB

Per la realizzazione di questo job abbiamo utilizzato due processi di MAP/REDUCE, il primo per trovare per ogni prodotto il numero complessivo di pezzi venduti, e il secondo per ordinare i prodotti in ordine decrescente.

Di seguito riportiamo l'implementazione in pseudocodice:

MAP

```
function map(key, value, context){

    line: value
    //Tokenizziamo la linea del dataset per il simbolo ","
    token: TOKENIZE(line, ",")

    while (token.hasNext())
        word: token.next()
        //Identifichiamo l'elemento data della linea
        if(token.notContain("-"))
            //Aggiungiamo al contesto tutti prodotti della linea con un unità
    associata
            context: add(word, 1)
     end while

}
```

REDUCE

```
function reduce(key, value, context){

    product: key

    //sommiamo tutti le unità corrispondenti al prodotto
    FOR unit in values {

        somma =  somma +1

    }
```

```
        //Aggiungiamo al contesto il  prodotto con la relativa frequeza nel dataset
        context: add(product, somma)

}
```

SortMAP

```
//In imput a questa funzione va il risultato della reduce sopra descritta
function sortMapper(key, value  , context){

        line: value
        //Tokenizziamo la linea il simbolo 'spazio'
        token: TOKENIZE(line)

        //Il primo elemento che trova è il prodotto, che inseriremo come valore nel
        risultato del mapper
        word:  token.next()

        //Prendiamo la frequenza relativa al prodotto
        number:  token.next()
        context: add(number, word)

}
```

SortREDUCE

```
//Il reduce prende come input il risultato del comparatore (frequenza, prodotto)
function sortReducer(key, values [] , context){

        //In questo caso abbiamo un solo valore come values e corrisponde al
        prodotto, perciò ci basterà invertire la coppia
        for product in values
                context: add(value, key)

}
```

II JOB

Alla termine di questo job si ottiene si genera, per ciascun prodotto, l'andamento delle vendite del primo trimestre 2015. Per realizzarlo abbiamo utilizzato un processo di MAP/REDUCE

Di seguido riportiamo l'implementazione in pseudocodice:

MAP

Assunto che il dataset contiene acquisti relativi ad un anno specifico. In questo mapper generiamo una lista di coppie (data, prodotto) dei soli prodotti venduti nel primo trimestre.

```
function map(key, value , context){

    dataSoglia: 1/04/2015//E' la data che stabilisce il limite del trimestre
    line: value
    //Tokenizziamo la linea del dataset per il simbolo ","
    token: TOKENIZE(line, ",")
    //il dataset per primo elemento della linea ha una data, la memorizziamo nella
    variabile 'data '
    data: token.next()

    if data < dataSoglia //Se la data appartiene al trimestre
        while (token.hasNext())
            product: token.next()
            context: add(product,data)
        end while

}
```

REDUCE

Abbiamo come input un prodotto e la lista di date in cui è stato venduto, per ciascun prodotto generiamo il trend nel primo trimestre

```
function Reduce(key, values [] , context){

    line: value
    //Tokenizziamo la linea del dataset per il simbolo "spazio"
    token: TOKENIZE(line)
    product: token.next()
    dates[]: values //prendiamo la lista di date in cui è stato venduto il prodotto
    analizzato

    //conto quanto è stato venduto per ciascun mese
    for data in dates
        if data in month(1) //se la data è nel mese di gennaio
            trendGennaio++ //incremento vendita nel mese relativo
```

```
        if data in month(2) //se la data è nel mese di febbraio
            trendFebbraio++ //incremento vendita nel mese relativo
        if data in month(3) //se la data è nel mese di febbraio
            trendMarzo++ //incremento vendita nel mese relativo

    //creiamo una descrizione con data e relativo trend
    trend: '01/2015: ' + trendGennaio + '02/2015: ' + trendFebbraio +'01/2015: ' +
    trendMarzo

    //Aggiungiamo il prodotto con il trend al contesto
    context: add(product,trend)

}
```

III JOB

Alla termine di questo job si genera una lista 10 coppie di prodotti che vengono più frequentemente venduti con il relativo numero di occorrenze. Per realizzarlo abbiamo utilizzato un processo di MAP/REDUCE

Di seguito riportiamo l'implementazione in pseudocodice:

MAP

Assunto che ogni riga del dataset corrisponde a una data, generiamo allora tutte le possibili coppie della riga . In output il mapper ci darà un oggetto coppia prodotti e un unità

```
function map(key, value  , context){

    line: value
    //carichiamo un array di prodotti dividendo gli elementi per  la virgola
    products []: SPLIT(line, ',')

    //generiamo tutte le coppie di prodotti dall'array ricavato sopra
    couples []: ALL_COUPLES(products)

    //per ogni coppia inseriamo nel contesto la coppia e l'unità 1
    for couple in couples
        context: add(couple, 1)

}
```

REDUCE

Ora basta semplicemente sommare tutte le unità presenti in values[] e inserire nel contesto la coppia con relativa frequenza

```
function Reduce(key, values [] , context){

    //La coppia corrisponde alla chiave
    couple: key

    //sommiamo tutti le unità corrispondenti al prodotto
    for unit  in values
        somma++;

    //Aggiungiamo al contesto la coppia con la relativa frequeza
    context: add(couple, somma)

}
```