Concordia University
Department of Computer Science and Software Engineering
COMP 6521 - Advanced Database Technology and Applications.
Fall 2015

Assignment 1. Solution

---

1. (4.5)

$$F_{COUNT(title)}(\pi_{title}(\sigma_{director='Bergman'}(Movies)) \cap \pi_{title}(Pairscope))$$

or

$$F_{COUNT(title)}(\pi_{title}(\sigma_{director='Bergman'}(Movies) \bowtie Pairscope))$$

Returns 'yes' if the result > 0, otherwise 'no'

(4.6)

$$\pi_{d1,d2}(\sigma_{d1=a2 \wedge d2=a1}(\rho_{M1(t1,d1,a1)}(Movies) \bowtie \rho_{M2(t2,d2,a2)}(Movies)))$$

(4.7)

$$\pi_{director}(\sigma_{director=actor}(Movies))$$

(4.8)

$$\pi_{a1,actor}(\sigma_{title=t1 \wedge actor \neq a1}(\rho_{M1(t1,d1,a1)}(Movies) \bowtie Movies))$$

2. (a) i. $\pi_{bar}(Sells \bowtie \sigma_{drinker=Bill}(Likes))$

ii.
```
SELECT Sells.bar
FROM Sells,Likes
WHERE Sells.beer = Likes.beer
AND Likes.drinker = 'Bill';
```

(b) i. $\pi_{drinker}(Frequents \bowtie Sells \bowtie Likes)$

(ii.)
```
SELECT Frequents.drinker
FROM Frequents, Sells, Likes
WHERE Frequents.bar = Sells.bar
AND Sells.beer = Likes.beer
AND Likes.drinker = Frequents.drinker;
```

(c) i. $\pi_{drinker}(Frequents) - \pi_{drinker}(Frequents - \pi_{drinker.bar}(Sells \bowtie Likes))$

(ii.)
```
SELECT Frequents.drinker
FROM Frequents
WHERE Frequents.drinker NOT IN
(
SELECT F.drinker
FROM Frequents AS F
WHERE F.bar NOT IN
(
SELECT bar
FROM Sells, Likes
WHERE Sells.beer = Likes.beer
AND Likes.drinker = F.drinker
)
);
```

(d) i. $\pi_{drinker}(Frequents) - \pi_{drinker}(Frequents \bowtie Sells \bowtie Likes)$

(ii.)
```
SELECT Frequents.drinker
FROM Frequents
WHERE Frequents.drinker NOT IN
(
SELECT F.drinker
FROM Frequents AS F, Sells, Likes
WHERE F.bar = Sells.bar
AND Sells.beer = Likes.beer
AND Likes.drinker = F.drinker
);
```

3. (a) **The First Map Function:** For each integer i in a chunk, produce the key-value pair $(chunk\_id, i)$.

   The First Reduce Function: Find the tuple with maximum value; i.e. output the key-value pair $(chunk\_id, max(i_1, i_2, i_3, \ldots, i_n))$.

   The Second Map Function: For each maximum integer of every chunk i, produce the key-value pair $(0, i)$. Note that there is only one reducer to compute the maximum of all maximums obtained from each reducer.

   The Second Reduce Function: Simply produce the key-value pair $(0, max(max_1, max_2, max_3, \ldots, max_n))$ which is the final output.

   (b) **The First Map Function:** For each integer i in a chunk, produce the key-value pair $(chunk\_id, i)$.

   The First Reduce Function: Find the average of all values; i.e. output the key-value pair $(chunk\_id, [avg(i_1, i_2, i_3, \ldots, i_n), k])$ where k is the number of integers in the reducer.

   The Second Map Function: For each average integer of every chunk i and the number of integers in it k, produce the key-value pair $(0, [i, k])$. Note that there is only one reducer to compute the weighted average of all averages obtained from each reducer.

   The Second Reduce Function: Simply produce the key-value pair $(0, avg(avg_1 \times k_1, avg_2 \times k_2, \ldots, avg_n \times k_n))$ which is the final output.

   (c) **The Map Function:** For each integer i in a chunk, produce the key-value pair $(i, 1)$.

   The Reduce Function: Associated with each key i there will be one or more values. Produce output $(i, 1)$ in every case.

   (d) **The First Map Function:** For each integer i in a chunk, produce the key-value pair $(i, 1)$.

   The First Reduce Function: Produce the number of distinct integers inside the reducer and output the key-value pair $(0, k)$ where k is the number of distinct integers.

   The Second Map Function: For every key-value pairs $(0, k)$, produce $(0, k)$ as it is.

   The Second Reduce Function: Produce the key-value pair $(0, \Sigma_{i=0}^{n-1} k_i)$ as output.

4. (a) *Bag Union*

The Map Function: Creates a key-value pair for each tuple t in either R or S, say $(t, 1)$. Group by key forms $(t1, [1, ..., 1])$.

The Reduce Function: Combines the values associated with each key t, so reads each $(t, [1, 1, ..., 1])$ and then aggregates each key-value pair into (t, sum). The output would be $(t_1, sum_1), (t_2, sum_2), \ldots$.

(b) *Bag Intersection*

The Map Function: Turns each tuple t from R into a key-value pair $(t, t(R))$ and tuple t from S is $(t, t(S))$. So we get $(t, (R, 1))$, $(t, (R, 1))$ and also for $(t, (S, 1)), \ldots$ and so on.

The Reduce Function: Gets each key-value pair and sum the count of R and S for each tuple t, say $(t, [(R, sum), (S, sum)])$, then it outputs the minimum sums, so the output would be $(t, min(sums))$. If the input has only one element like $(t, (R, sum))$ or $(t, (S, sum))$, then the output would be nothing.

(c) *Bag Difference*

The Map Function: Turns each input tuple t in R into a key-value pair $(t, 1)$ and also for every input tuple t in S, turn it into a key-value pair $(t, ???1)$, so group by key forms $(t, [1, 1, ???1, 1, ???1, \ldots])$.

The Reduce Function: Gets each key-value pair $(t, [1, 1, ???1, 1, ???1, \ldots])$, then aggregates each key-value pair into (t, sum). The reducer will output only positive sums. Thus, the output would be $(t_1, sum_1), (t2, sum2), (t3, sum3), \ldots$ for $sum_1, sum_2, sum_3, \ldots > 0$.

5. Let S be a set of b-bit strings. To compute d-Hamming Distance over S, The idea is to divide the problem based on possible answers. Let s_1 and s_2 be two strings of Hamming Distance d. So, they will mismatch on exactly d bits. So, for every substring of s_1 and s_2 of length greater than d they will match on at least one bit. The algorithm divides each b-bit strings into $d+1$ substrings (here for simplicity we assume d divides b), and them will match in at least one of these $d+1$ substrings.

   Simply, the replication rate is $d+1$, since each bit-string is replicated $d+1$-times.

   The number of reducer needed is based on the number of $b/d+1$-substrings which can be obtained from all b strings. That is the algorithm needs $(d+1)2^{b/(d+1)}$ reducers.

   The Map Function: The input string s is divided into, $s_1, s_2, \ldots, s_{d+1}$. so we have:

   Key $= (i, s_i)$ for $i \in \{1, 2, \ldots, d+1\}$

   Value $= s$

   The Reduce Function: Every reducer receives a set of bit-strings with the same key, and check their values to see whether they are of Hamming Distance d.