Homework 3

CS386D Database Systems
Created on 2/13/20
Version: 1.0

Assigned: 2/13/20

Due: 2/19/20, 11:59PM, electronically on canvas

Objectives: There are three parts of the assignment,

A. Bitmap indexing
B. Bloom filters
C. Finer points of SQL and relational algebra (i.e. sets, bags and nulls), and the extended relational operators.

Reading and discussion:

A. Bitmap indexing, Ch. 14.7 All in the text.

B. Bloom Filters, The Wikipedia page, https://en.wikipedia.org/wiki/Bloom_filter (recall this material is central to contemporary database developments. Despite being first published in 1970, it is not in the textbook). Also you'll see the Wikipedia page covers a number of variations/improvements on Bloom filters not covered in lecture. Though it won't be on an exam in this class, if you think your future may include low-level visibility (i.e. source-code) with a parallel cloud-native database, you should familiarize yourself with the full contents of the Wikipedia page. Also, though not uploaded to Box, supplementary reading is Bloom's original paper. It is linked into both the lecture notes and the Wikipedia page.

C. Reading, text, 5.1, 5.2, 16.1 and 16.2.

The Assignment:

Part A: Text problems 14.7.1, 14.7.3a

Part B: Consider the Bloom filter presented in Jason Davies' interactive demo, *http://www.jasondavies.com/bloomfilter/*

Before you get started, you should play with the demo until you are comfortable that the idea actually works.

1. For the following, be careful to use all lowercase letters. Start from an empty (reset) demo filter. Place the letter 'z' in the query box. (the box on the right side of the diagram). Start from 'a', add the alphabet, as keys, to the filter. (i.e. add a, then b, then c and so on).
   a. How much of the alphabet could you add as keys to the filter before getting a false positive hit? If you are at all concerned about your answer, proceed to question 2.

   b. If I asked you to repeat this problem with 5 letter words would you expect different results?

2. For the demo Bloom filter,
   a. If one key is stored in the filter, e.g., ‘z’. What is the probability of a false positive?
   b. If 20 keys are stored in that Bloom Filter, what is the probability of a false positive?
   c. From an empty filter, add as keys the letters a through t, (20 keys). Type your name in the query box. Did the filter report your name is probably there (True), or did the filter report your name was definitely not there, (False). There will be a Piazza poll. Enter your answer, (True or False) in the Piazza poll.

3. Suppose Jason Davies was given a set of requirements for a Bloom Filter, detailing the size (length in bits) of the Bloom Filter, and that he should use the optimal number of hash functions. The result was precisely the Bloom Filter illustrated on the web site. How many keys did the requirements document state were to be stored?

Part C.

1. Text problem: 5.1.1, 5.1.2, 16.2.2 b, c

   Think about 5.1.4[1]

2. Consider the following pair of tables, R and S.

R

thePrimaryKey	name	joinKey1
1	Andrea	101
2	David	Null
3	David	Null
4	Dan	Null
5	John	106

S

thePrimaryKey	romanNumeral	joinKey2
6	V	101
7	X	Null
8	L	105

Create a database on your machine for tables R and S and load the 8 rows.
Consider each of the following queries in relational algebra. Translate the query to SQL and execute the query on the database. When translating, do the easy, obvious translation without concern for set vs. bag semantics. i.e. DO NOT introduce the SQL distinct

[1] *Think about* problems are just that. Read the problem and understand the issue. Optionally you may do the problem on paper if that helps you digest the material. In either case, do not hand anything in. It will not be graded. Post on Piazza if you get stuck.

operator. The goal of the homework is to see how Null is treated. (Note: I've never located a butterfly/join symbol in MS-Word. So in the relational algebra expression below, the word "join" replaces the butterfly symbol.)

i. $\sigma_{\text{joinKey1} = 101}(R)$

ii. $\sigma_{\text{joinKey1} \neq 101}(R)$

iii. $\sigma_{\text{joinKey1} = \text{Null}}(R)$

iv. $\Pi_{\text{Name, joinKey1}}(R)$

v. $\Pi_{\text{joinKey1}}(R)$

vi. $R \text{ join}_{\text{joinKey1} = \text{joinKey2}} S$

vii. $R \text{ join}_{\text{joinKey1} \neq \text{joinKey2}} S$

viii. $R \text{ full outer join}_{\text{joinKey1} = \text{joinKey2}} S$

ix. $R \text{ left outer join}_{\text{joinKey1} = \text{joinKey2}} S$

x. $R \text{ left Semijoin}_{\text{joinKey1} = \text{joinKey2}} S$

xi. $R \text{ left Antijoin}_{\text{joinKey1} = \text{joinKey2}} S$

For each part of question C2, turn in only the output rows, clearly separating and labeling the output

3. For parts vi and vii, write three different equivalent SQL queries that implement the query shown in relational algebra. (That is the 1 you wrote above, plus 2 more. You should execute them to test if all three produce the same result.)
4. For parts viii through xi, write two different equivalent queries that implement the query shown in relational algebra.