# Homework 4 V2.0

CS386D Database Systems
Created on 2/24/20
Version: 2.0

**Assigned:** 2/23/20

**Due**: 2/28/20, Friday11:59PM, electronically on canvas

**Objectives:** Additional exercises per contemporary storage management

## Storage Model Summary

| Model | Data Storage | Primary Index | Secondary Index |
|---|---|---|---|
| Conventional RDBMS | Rows | B+ tree | B+ tree, plus vendor specific additions |
| Column Store (RDBMS) | Compressed Bit Maps | N//A | ? |
| Key Value (NoSQL) | Hash partitioned, replicated, large compressed pages | Bloom Filter | Bloom Filter, maybe others |

Relational databases and B+trees are well covered in the text and past homeworks, thus:

**The Assignment:**

**1. Continuation of bitmap indexes, exercise 14.7.3, as they pertain to column stores.**

**Correction:** This problem concerns compressed bitmaps. The objective is to compare the storage needed by a column store to the storage needed for a conventional RDBMS. Thus, the question is a continuation of 14.7.3b, which was not assigned. (Makes it tough for this to be a continuation.). Use the following solution to 14.7.3b. This solution makes a simplifying assumption, to get it to you quickly.

Sol'n 14.7.3b

The problem states that a value appears, in round robin fashion every m rows. Ignoring the first run, (the first 1 in the bitmap), there will be a run of m-1 zero's and a 1repeated many times. Such a run will require $2\lceil \log_2(m-1) \rceil$. Given there are m different values, and 1,000,000 rows, each value will appear ~1,000,000/m times, meaning the run will

repeat that many times. Thus the total number of bits is, approximately

$m * 2\lceil \log_2(m-1) \rceil * (1{,}000{,}000/m) = 2{,}000{,}000 \lceil \log_2(m-1) \rceil$

This solution ignores the easy to identify collection of m different runs that precede the regularly sized (round robin) runs, (1, 01, 001, …, m-1 zeros 1). Related to that I have not properly assessed if there is a fencepost problem. But these are second order effects such that the above is sufficient for you to do the homework and achieve its learning objective.

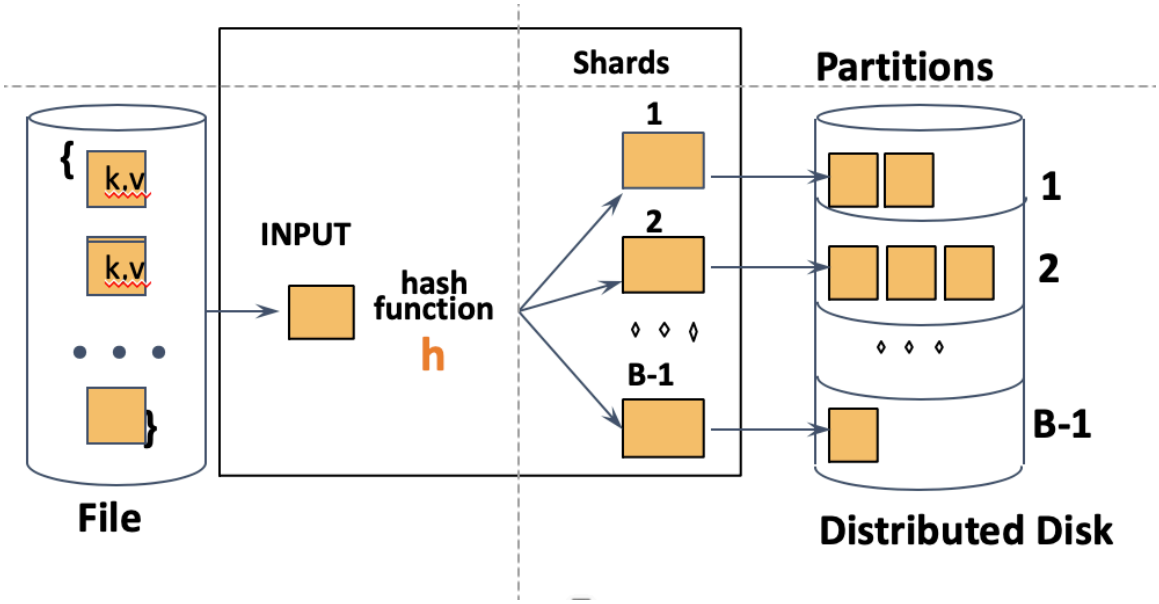
---

Consider two tables, S and T, of 1,000,000 rows and 100,000,000 rows respectively. As may prove convenient, n is the number of rows in a table. Further properties of S and T are identical starting with, S and T have 101 columns, including a column of type integer that serves as the primary key. Let the columns be named $c_i$, $i = 0 .. 100$. Similarly, let $m_i$ be the number of different values in column $c_i$. Assume all the following questions are in the context of storing either S or T, in their entirety in a column store database. Recall from lecture, that means data values are not stored directly, but rather each data value, (per column), is represented by a pair, (*data-value*, compressed-*bitmap)*. E.g. ("Austin", 1011110100….). Let et $c_0$ be the primary key. Clearly, $m_0 = n$

a) Develop an expression that, given all $m_i$, will determine the number of bytes needed for all the bitmap indexes to needed to store T (just the bitmaps).
b) Does your answer to part a need to include a bitmap index for $c_0$. Hint: review the textbook per the implementation of bitmap indexes.
c) Suppose, for all columns i, $i \neq 0$, the number of values in half of the columns is $m_i = n/1000$ and the other half, $m_j = 10{,}000$. How many bytes will be required to the bitmaps for i) S, ii) T?
d) Suppose data, $c_0$, requires 4 bytes, the columns where $m_i = n/1000$ required an average of 25 bytes, and the columns st. $m_j = 10{,}000$ require and average of 20 bytes. In a conventional RDBMS (row store) how many bytes is required to store just the data table i) S, ii) T. If data pages contain 4kbytes, how many data pages are required to store iii) S, iv) T

## 2. Loading and filtering a key, value store.

Consider loading a large file into a cloud-native key, value store. Figure from course notes repeated below.




The storage system will hash each key to an integer. Servers are logically numbered, and the hash result determines which server is responsible for storing the key value pair[1]. At the data level, the contents of each server is commonly referred to simply as a *shard*. It is not clear if shard and partition mean different things. That said, in most common usage, within each shard there is still a notion of disk block, a unit of transfer between the server and its disks. But compared to the relational literature where page and block are used interchangably, in these contemporary distributed databases a unit of storage is most commonly called a *block*.

Sharding assures that all keys, in all blocks in a server will contain values whose key hashed to the same server. Blocks are large. In Cloudera's standard release of HDFS the administrator may choose 64Mbytes or 128Mbytes as the block size. Also, recall blocks are compressed before they are stored on disk. Compression ratios of 10 to 100 can be expected.

Suppose it is your job to (optimally) configure a Bloom filter-based index for the blocks of a Cloud database so that individual blocks are fetched only if the key appears in the Bloom filter.
Assume,

- The database is 8Tbytes (8 terabytes = 8 x $2^{40}$).
- The logical partitioning of the database is to 128 shards. i.e. there are 128 servers. For this problem we will ignore that shards are replicated on different servers to support fault tolerance, (durability).
- Suppose, on average, a key value pair requires 1024 bytes of storage. (keys must be stored with the values. Why?)

[1] Actually, since storage is replicated for fault tolerance (an implementation of durability), the hash value determines a set of servers, not a single server, that will store redundant copies of the data. For this question, assume there is no redundancy, and thus no fault tolerance. It won't be important.

- Let blocks be 64Mbytes.

a) Why must keys be stored with the values?

b) Have the Bloom filter use 10 bits per element expected to be stored.
   i. How many bits will be required for each filter?

   ii. How much memory, on average, will be required, per server, to represent the Bloom filters?

   iii. What is the optimal number of hash functions?

   iv. What is the probability of a false positive?