

Bloom Filters

$\leq n$ elements in a set

m bits to represent it $m \ll n$

$(1) \dots (n-1) (n)$

exact set

membership queries:

1

0

ADD

$ISMEMBER(x, S)$ n bits
for a set of size n

(inexact set membership):

ADD(item, S')

$ISMEMBER(x, S')$

if $x \in S$ then report TRUE

but can also return MAYBE

ADD(Scooby, S')

ADD(Velma, S')

$ISMEMBER(Velma, S) = \text{TRUE}$ true positive

$ISMEMBER(Scooby, S') = \text{TRUE}$ false positive

$ISMEMBER(x, S') = \text{FALSE}$

definitely true

ADD(): notary
ISMEMBER: true

One bit Bloom Filter

0

$h(x) = 0$ if x is even
 $= 1$ if x is odd

bit ←

ADD(x, S) $\Rightarrow h(x)$ OR bit

ADD(20, S)

ADD(40, S)

ISMEMBER(20, S) - MAYBE

ISMEMBER(x, S) $h(x)$

ISMEMBER(40, S) -
MAYBE

— Same - MAYBE

ISMEMBER(60, S) -
MAYBE

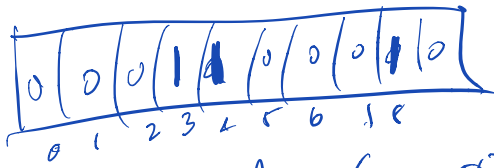
ISMEMBER(41, S) - NO

1

ADD(51, S)

16 bits

$$h(x) = x \% 16$$



ADD(51, S)

ADD(40, S)

ADD(20, S)

k hash function, ~~ADD(17, S)~~ (smasher(17))

$$(1 - e^{-kn/m})^k \quad \text{false positive rate}$$

1 hash fn. $\frac{16 \text{ bits}}{n}$ $\frac{1000 \text{ elements}}{m}$

1.5% false positive rate

2 \rightarrow 0.1%

Bloom filter — insertion ✓

lookups ✓

deletion ✗

two BF insertion/deletes

Counting Bloom filter

Invertible Bloom Lookup Table
[Count from user]

trading ^{reduced} space & time
for accuracy

ADD DELETE FIND

Bloom filter — set membership

Set cardinality query

$|S|$

ADD

disk count

pay-per-disk
pay-per-conversion

CLICK FRAUD

mesothelioma

\$320

lawyers

Cardinality queries

Redis

data structure server

strings

hashes

lists

bitmaps

sets

sorted sets

| range query



geospatial indexes

— radius query

Bloom filters

HyperLogLog

— unique things
in a set

$$| \text{devire} | + | \text{os} | -$$

$$| \text{devire} \cup \text{os} | = | \text{devire} \cap \text{os} |$$

$$| A \cap B | = | A | + | B | - | A \cup B |$$

Count unique values \rightarrow (# unique elements)



$$\sim \quad w- \quad -$$

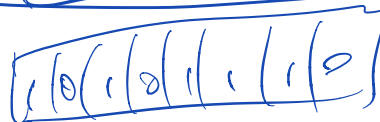
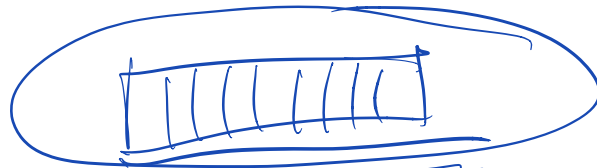
$$\& \quad 2 \quad 2$$

$$\text{hash}(\sim) ++$$

$$86115$$

~~1000~~

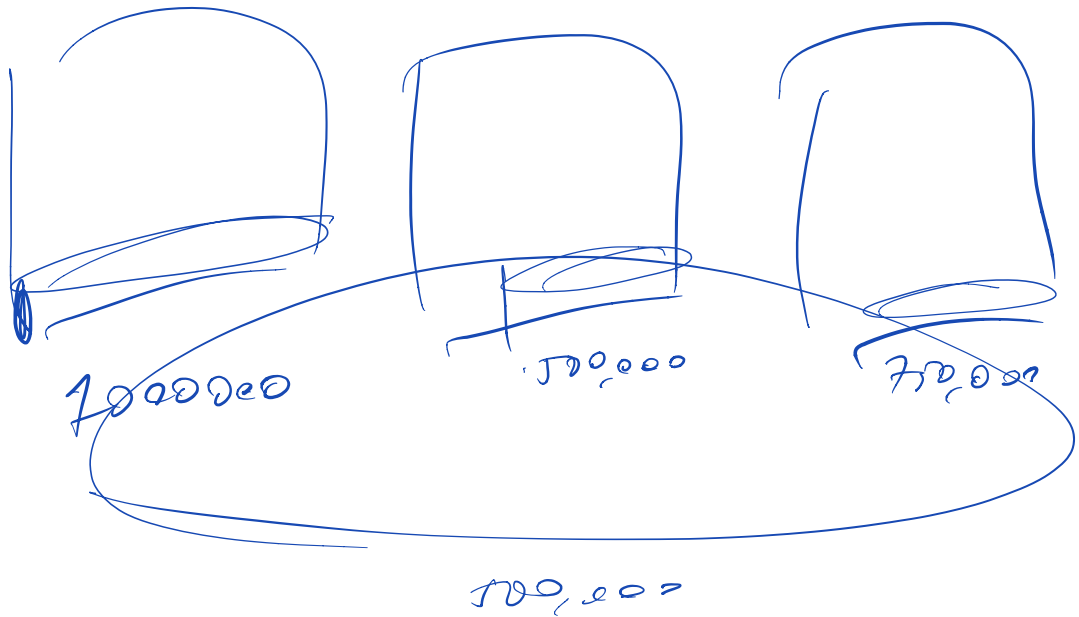
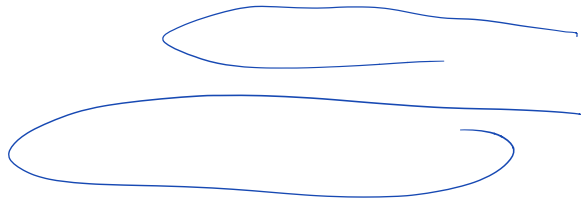
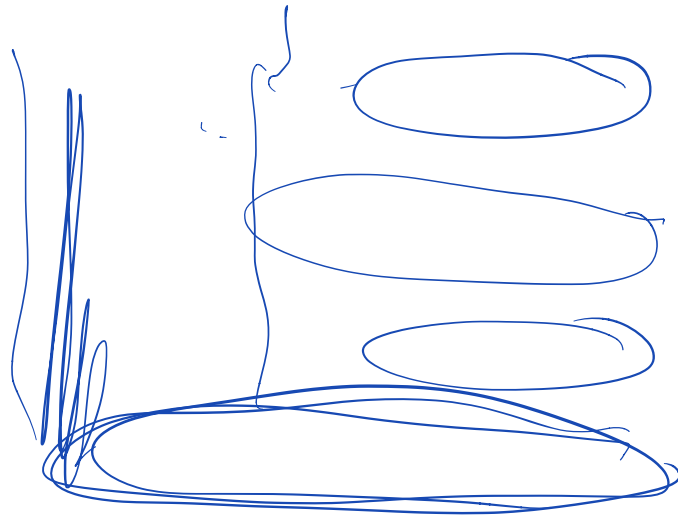
1... 000000



$\frac{1}{2}$ 0

$\frac{1}{4}$ 00

$\frac{1}{8}$ 000



$$\frac{1}{2^k} \approx k \text{ elements}$$

(with 64)



MongoDB

Fair — RAM

Sacrifice consistency guarantees

DB — schemas

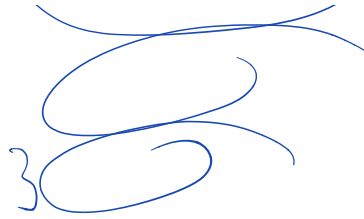
flat

relational

JSON



nested dictionaries
of documents



} 

{ key : value,
key : { k₁ : value
k₂ : value
:
}

Binary - Tree

posts.insert()
posts.insert-many()

Collection-oriented
Search for matching JSON key values
unique-id

partially "lazy"