Towards Managing Complex Data Sharing Policies with the Min Mask Sketch

Stephen Smart & Christan Grant

IRI 2017

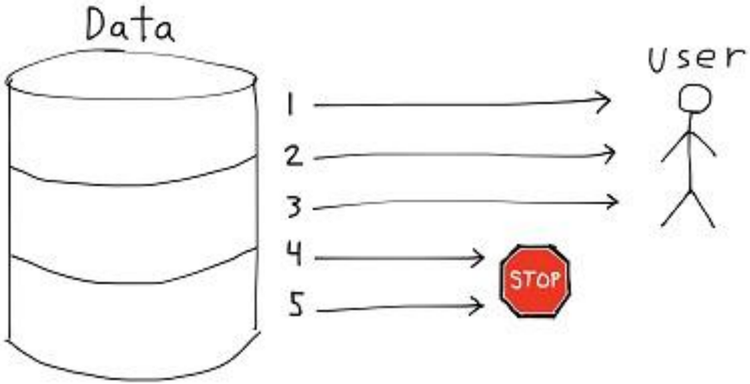
What are data sharing policies?

What are data sharing policies?

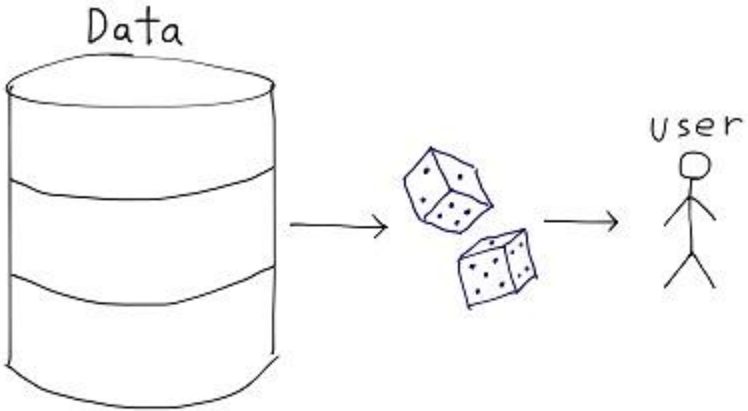
- A sharing policy is a set of expressions that describe how, when, and what data can be accessed.
- Examples:
    - ACL’s
    - IAM (Amazon Web Services)
    - Friend-based sharing
    - BitTorrent / Distributed data networks
    - Advertisements

What are simple data sharing policies?

A **single** expression describes how to share the data.



`LIMIT = 10`



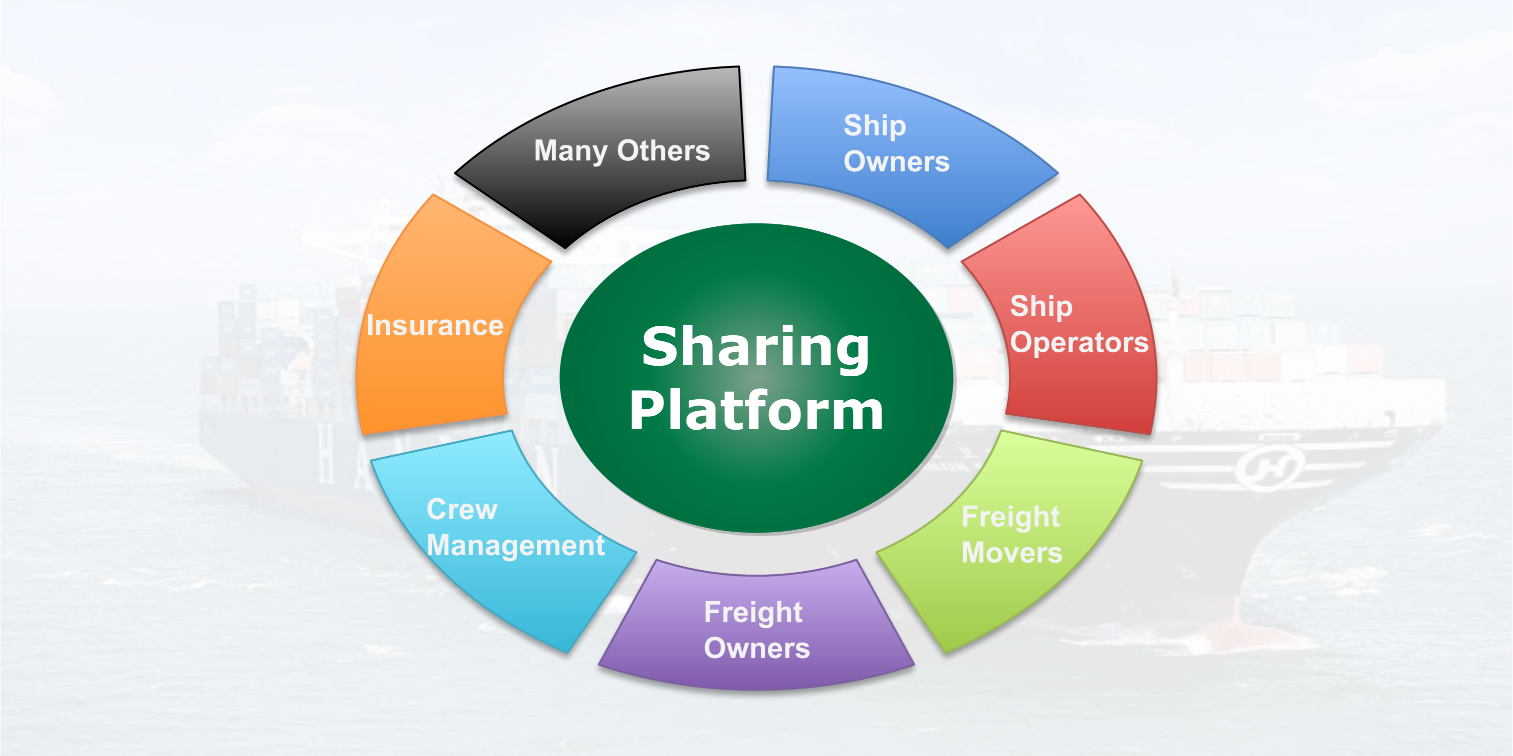
`random() < 0.167`

What are complex data sharing policies?

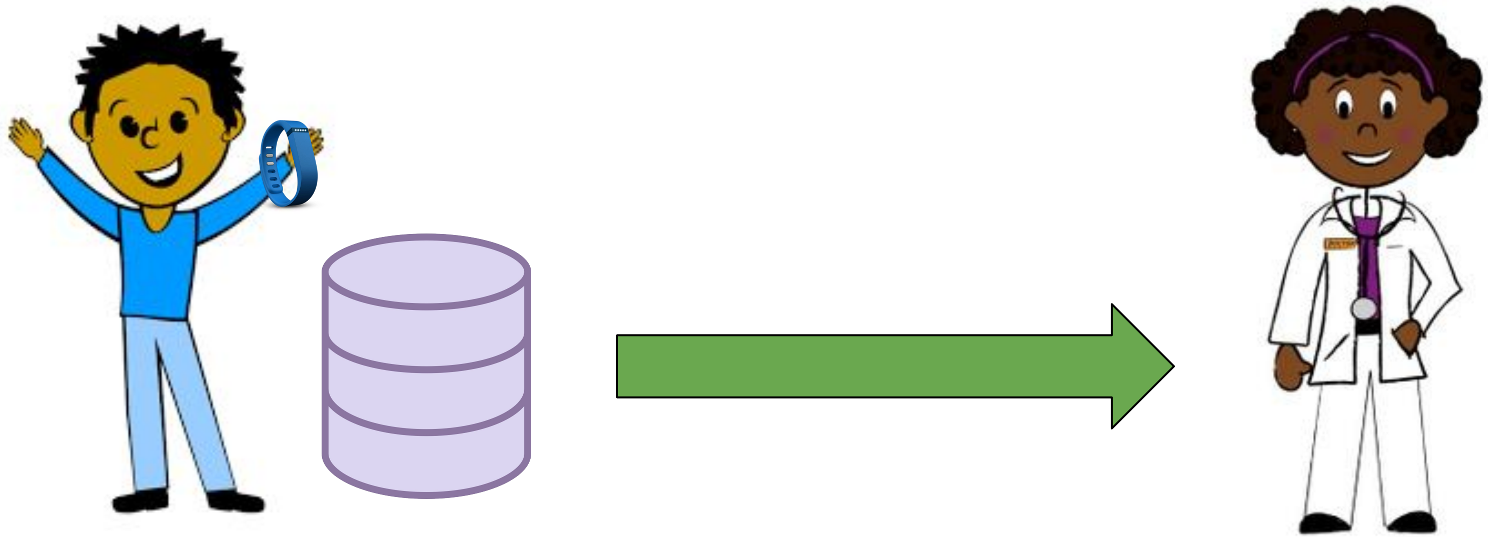
Multiple expressions describe how to share the data.

Sharing Policy ID(s)	Data
1	Record 1
3	Record 2
2	Record 3
1, 3	Record 4
1, 2, 3	Record 5

Example: Weather Company X



Example: Health Tracker Pro



Example Data Set

time	heart_rate	blood_sugar	body_temp
2016-02-20 04:05:06	71	95	98.6
2016-02-20 04:05:09	72	96	98.7
2016-02-20 04:05:09	72	94	98.7
⋮	⋮	⋮	⋮
2016-02-21 11:14:40	115	125	99.3
2016-02-21 11:14:43	115	124	99.5
2016-02-21 11:14:46	116	124	99.6

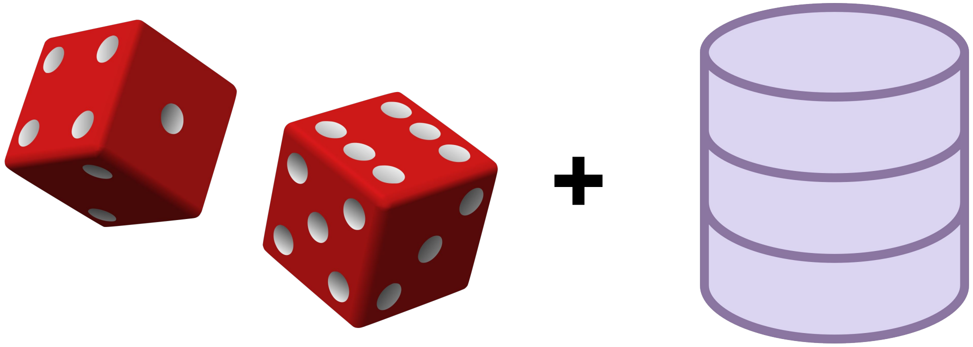
Example Data Set with Sharing Policies

time	heart_rate	blood_sugar	body_temp	high_hr	low_bs	high_bt
2016-02-20 04:05:06	71	95	98.6	0	1	0
2016-02-20 04:05:09	72	96	98.7	0	1	0
2016-02-20 04:05:09	72	94	98.7	0	1	0
⋮						
2016-02-21 11:14:40	115	125	99.3	1	0	1
2016-02-21 11:14:43	115	124	99.5	1	0	1
2016-02-21 11:14:46	116	124	99.6	1	0	1

How can we store this policy metadata more efficiently?

Probabilistic Data Structures

- Sacrifice a small amount of accuracy in exchange for space efficiency.
- Can answer queries about the data without needing to store the entire data set.
- Examples
  - Bloom Filter
  - Count Min Sketch



Bloom Filter

- Probabilistic data structure that is used to test whether an element is a member of a data set.
- Uses an array of bits and a collection of hash functions.
- Conceived by Burton Howard Bloom in 1970.

How Does it Work?

- Initialization:

Bloom Filter

How Does it Work?

- Initialization:
  - Set each bit in the array to 0.
  - Create k hash functions using technique from Kirsch et. al 2005

Bloom Filter

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bloom Filter: Inserting

- Insert an element, X.
- Let $k = 3$

Bloom Filter

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bloom Filter: Inserting

- Insert an element, X.
- Let *k* = 3
  - $h_1(X) = 7$

Bloom Filter

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bloom Filter: Inserting

- Insert an element, X.
- Let k = 3
  - $h_1(X) = 7$
  - $h_2(X) = 2$

Bloom Filter

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bloom Filter: Inserting

- Insert an element, X.
- Let k = 3
  - $h_1(X) = 7$
  - $h_2(X) = 2$
  - $h_3(X) = 11$

Bloom Filter

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bloom Filter: Inserting

- Insert an element, X.
- Let k = 3
  - $h_1(X) = 7$
  - $h_2(X) = 2$
  - $h_3(X) = 11$
- Each hash value corresponds to an index in the array of bits.

Bloom Filter

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bloom Filter: Inserting

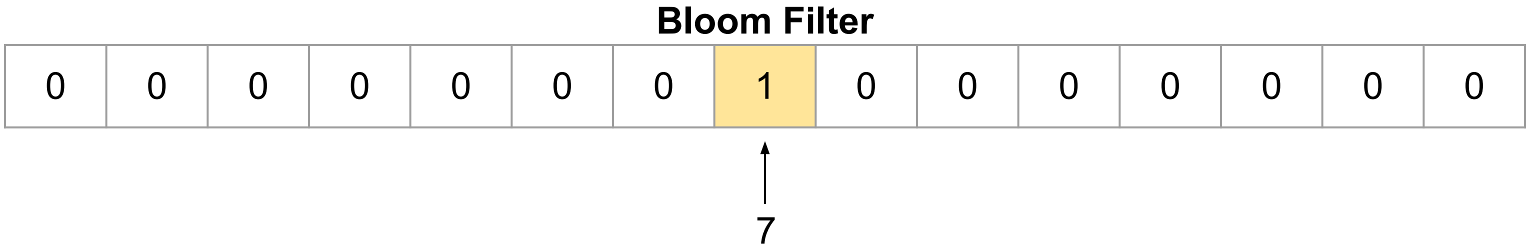
- Insert an element, X.
- Let k = 3
  - $h_1(X) = 7$
  - $h_2(X) = 2$
  - $h_3(X) = 11$
- Each hash value corresponds to an index in the array of bits.
- For each index calculated above, set the associated bit to 1.

Bloom Filter

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bloom Filter: Inserting

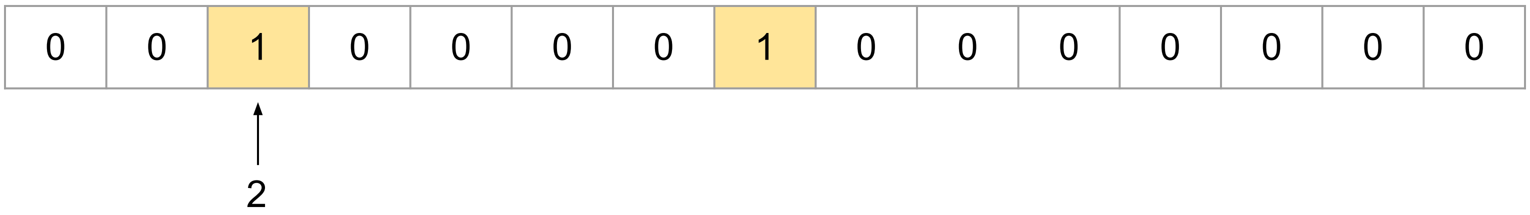
- Insert an element, X.
- Let k = 3
  - $h_1(X) = 7$
  - $h_2(X) = 2$
  - $h_3(X) = 11$
- Each hash value corresponds to an index in the array of bits.
- For each index calculated above, set the associated bit to 1.



Bloom Filter: Inserting

- Insert an element, X.
- Let k = 3
  - $h_1(X) = 7$
  - $h_2(X) = 2$
  - $h_3(X) = 11$
- Each hash value corresponds to an index in the array of bits.
- For each index calculated above, set the associated bit to 1.

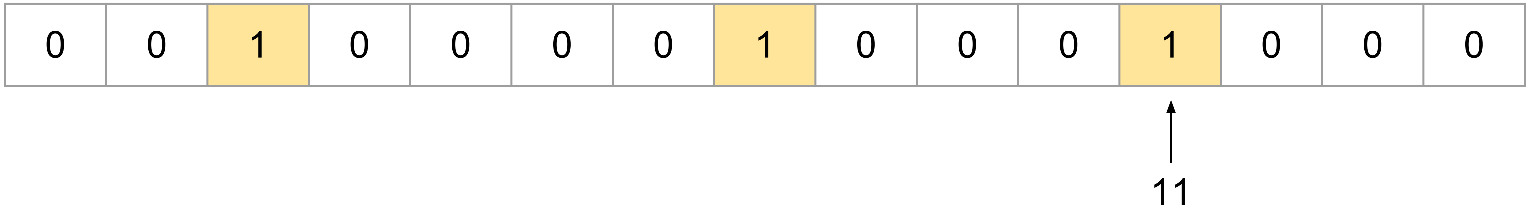
Bloom Filter



Bloom Filter: Inserting

- Insert an element, X.
- Let k = 3
  - $h_1(X) = 7$
  - $h_2(X) = 2$
  - $h_3(X) = 11$
- Each hash value corresponds to an index in the array of bits.
- For each index calculated above, set the associated bit to 1.

Bloom Filter



Bloom Filter: Querying

- Query an element, W.

Bloom Filter

0	0	1	0	0	0	0	1	0	0	0	1	0	0	0

Bloom Filter: Querying

- Query an element, W.
- Hash W using all *k* hash functions.

Bloom Filter

0	0	1	0	0	0	0	1	0	0	0	1	0	0	0

Bloom Filter: Querying

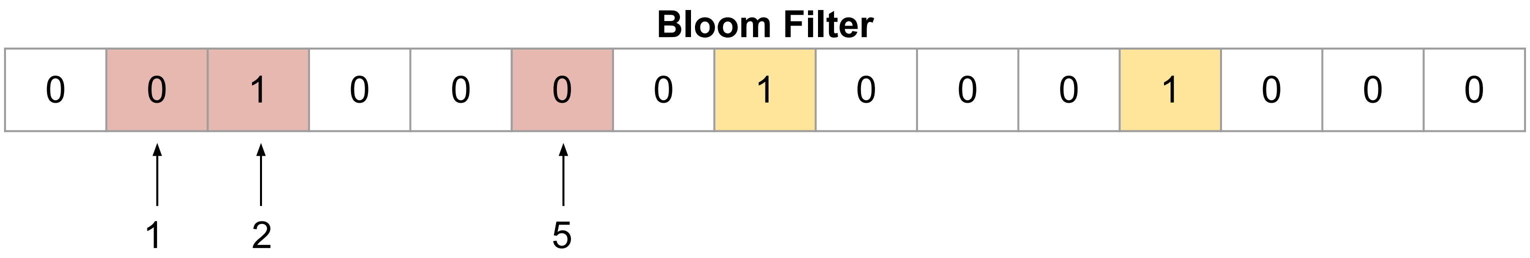
- Query an element, W.
- Hash W using all *k* hash functions.
    - $h_1(W) = 5$
    - $h_2(W) = 2$
    - $h_3(W) = 1$

Bloom Filter

0	0	1	0	0	0	0	1	0	0	0	1	0	0	0

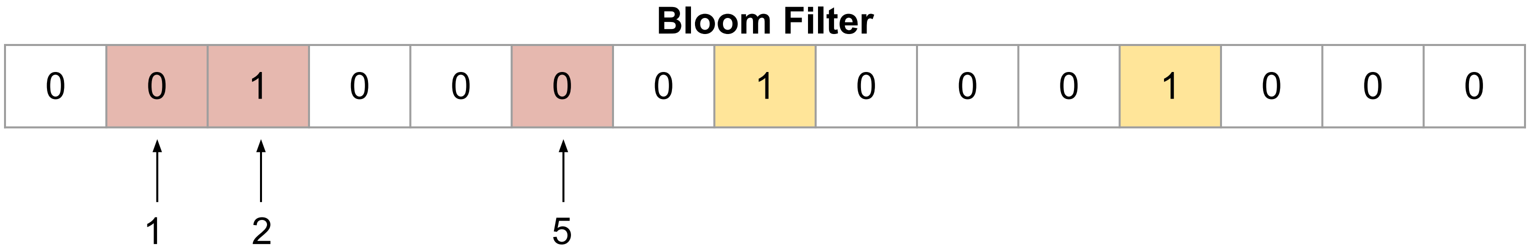
Bloom Filter: Querying

- Query an element, W.
- Hash W using all *k* hash functions.
  - $h_1(W) = 5$
  - $h_2(W) = 2$
  - $h_3(W) = 1$



Bloom Filter: Querying

- If all bits are 1, W is said to exist in the set.
- If all bits are **not** 1, W is said to not exist in the set.



Bloom Filter: False Positives

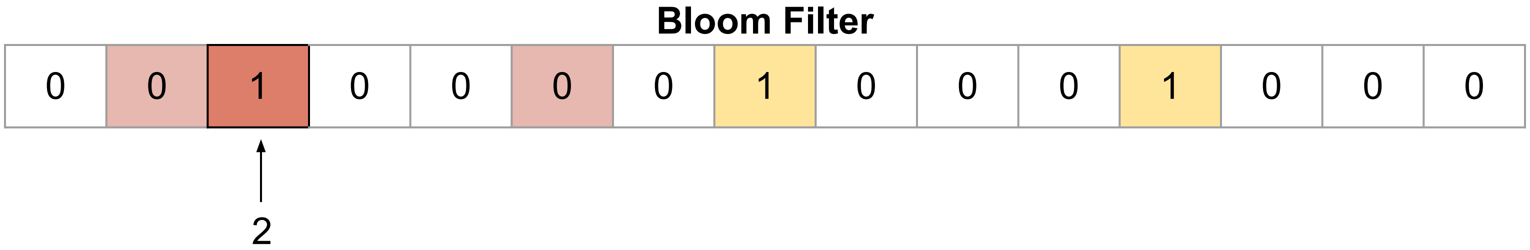
- Hash collisions can result in false positives.

Bloom Filter

0	0	1	0	0	0	0	1	0	0	0	1	0	0	0

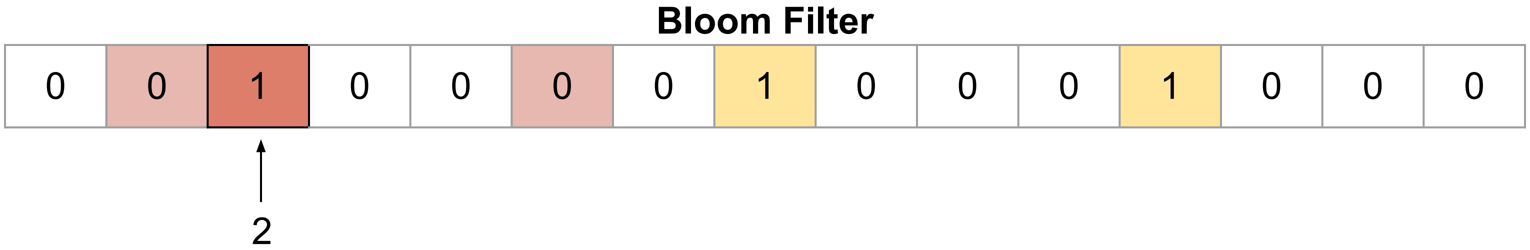
Bloom Filter: False Positives

- Hash collisions can result in false positives.
- $h_2(W)$ collided with $h_2(X)$



Bloom Filter: False Positives

- Hash collisions can result in false positives.
- $h_2(W)$ collided with $h_2(X)$
- If the result of all *k* hash functions collided with any other element, all the bits would be 1, even though W is not an element in the data set.



Bloom Filter: False Negatives are Not Possible

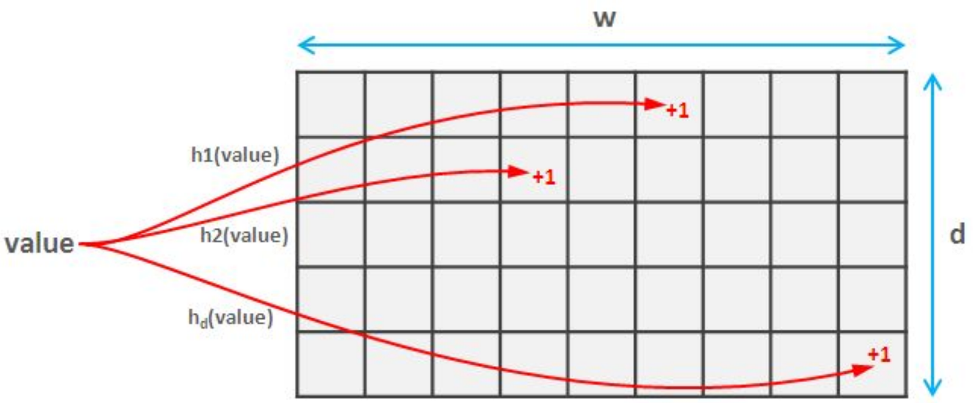
- If an element exists in the data set, the Bloom Filter query will always return true.

Bloom Filter

0	0	1	0	0	0	0	1	0	0	0	1	0	0	0

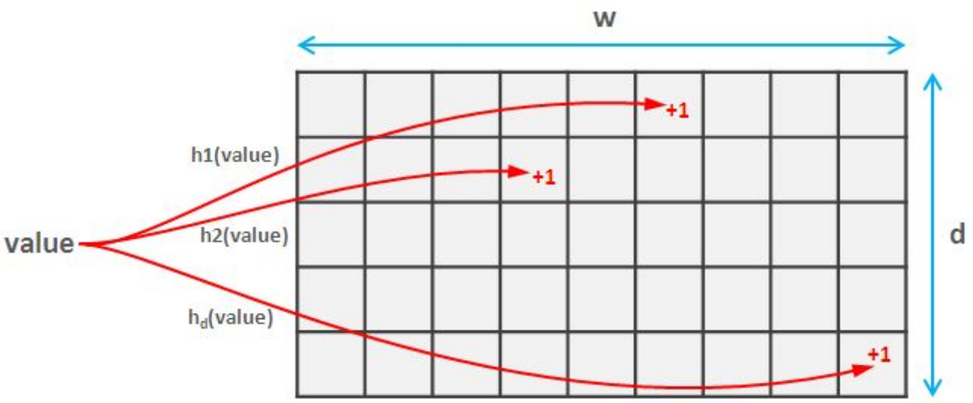
Count-min Sketch

- Like a Bloom Filter but uses an array of counters instead of an array of bits.
- Used to determine an element's frequency within a data set.
- Cormode et al. (2005)



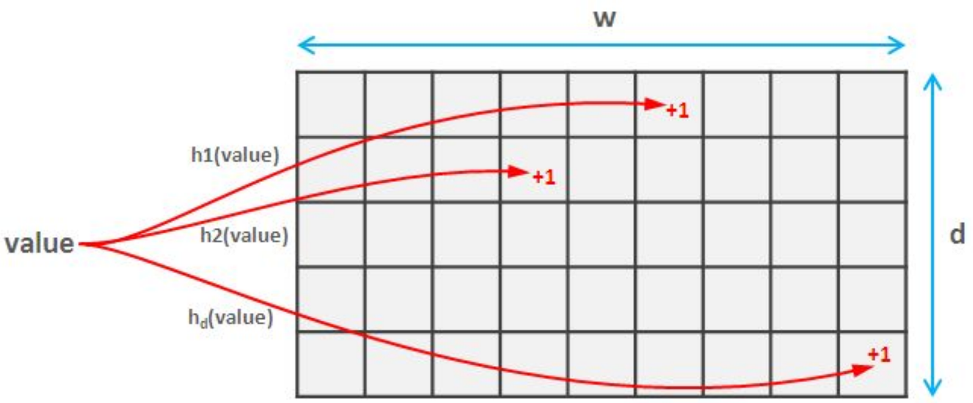
Count-min Sketch: Inserting

- When inserting an element, the element's primary key is hashed using all ***d*** hash functions.
- The counter value at each index is then incremented.



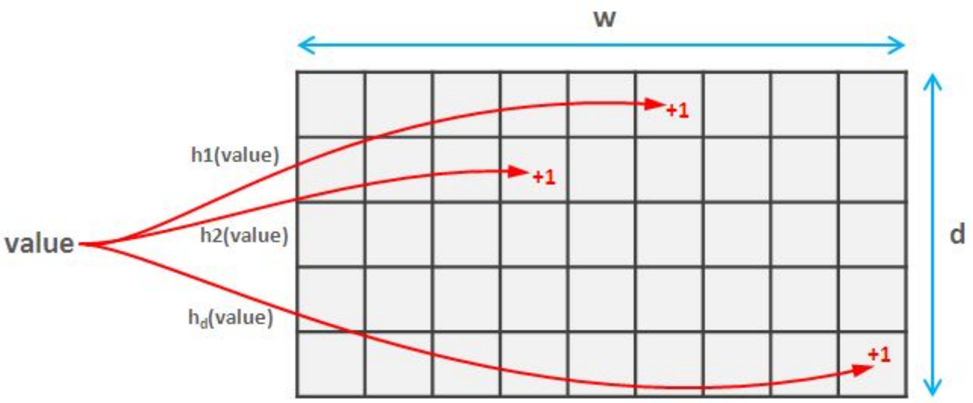
Count-min Sketch: Querying

- When querying an element, the element’s primary key is hashed using all ***d*** hash functions.
- The minimum counter value at each index is returned as the estimated frequency for the element.



Count-min Sketch: Frequency Estimates

- The frequency can be overestimated due to hash collisions.
- The frequency cannot be underestimated.

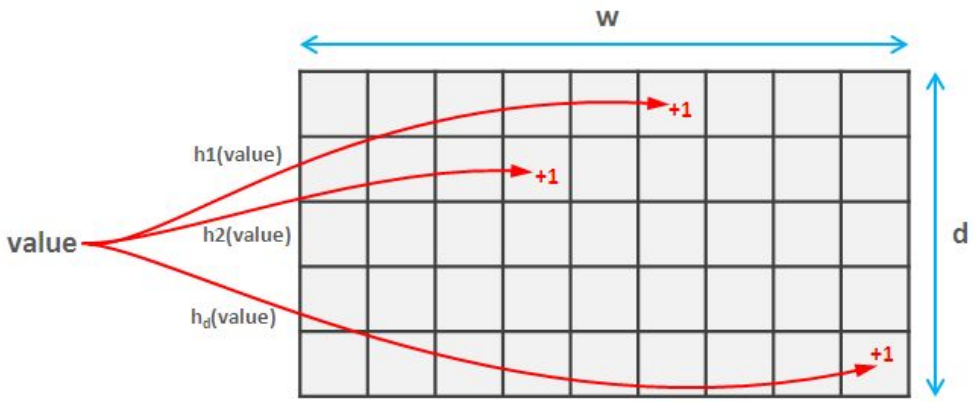


Count-min Sketch: Parameters

- Sketch is sized according to the desired quality.
- The frequency estimate is bounded by an additive factor of **ϵ** with probability **c**.
- **ϵ** and **c** are chosen by the developer.

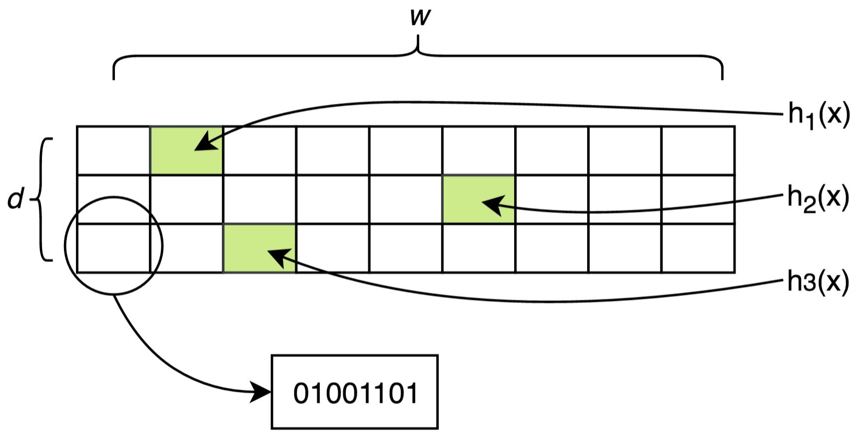
$$\mathrm{w} = \lceil \frac{e}{\epsilon} \rceil$$

$$\mathrm{d} = \lceil \ln(\frac{1}{1-c}) \rceil$$



Min Mask Sketch

- Like a Count-min Sketch but uses an array of bit strings instead of an array of counters.
- Used to determine an element's sharing policy information within a data set.
- This paper.



What Does the Bit String Represent?

- Each position in the bit string represents a possible expression to evaluate in order to share or restrict data.

Expression 1	heart_rate > 114
...	...
Expression 4	`random() < 0.167`
...	...
Expression 8	`LIMIT = 10`

```
00101001
```

What Does the Bit String Represent?

- Each position in the bit string represents a possible expression to evaluate in order to share or restrict data.
- If a bit at a particular position is set to 1, that expression is *active*

Expression 1	heart_rate > 114
...	...
Expression 4	`random() < 0.167`
...	...
Expression 8	`LIMIT = 10`

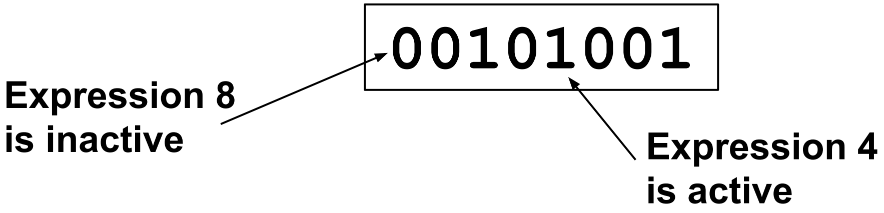
`00101001`

Expression 4 is active

What Does the Bit String Represent?

- Each position in the bit string represents a possible expression to evaluate in order to share or restrict data.
- If a bit at a particular position is set to 1, that expression is *active*.
- If a bit at a particular position is set to 0, that expression is *inactive*.

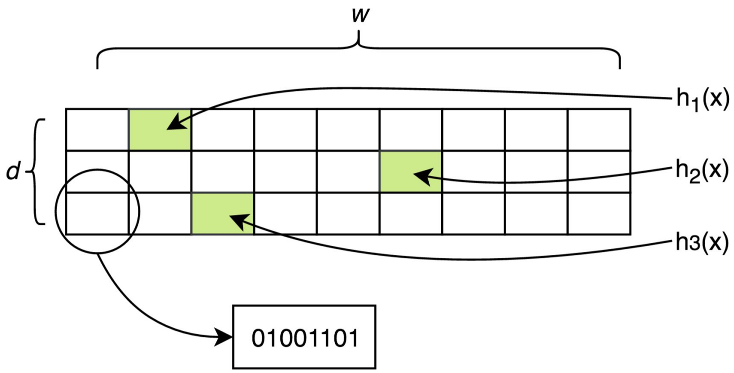
Expression 1	heart_rate > 114
...	...
Expression 4	`random() < 0.167`
...	...
Expression 8	`LIMIT = 10`



Min Mask Sketch: Inserting

- The new element is hashed based on its primary key (x) using the ***d*** different hash functions.

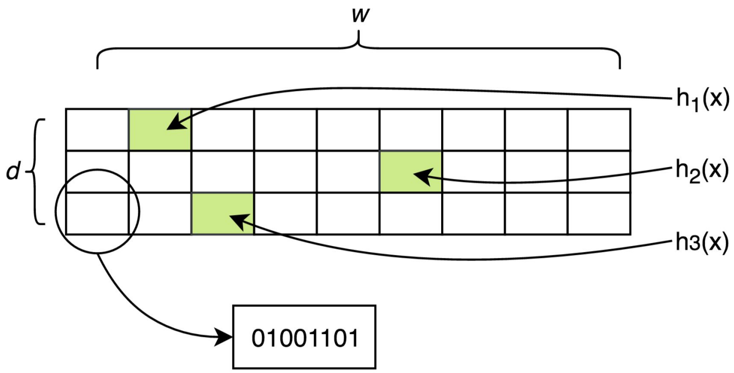
```
mms[h_i(primary_key)] |= policy_string
```



Min Mask Sketch: Inserting

- The new element is hashed based on its primary key (x) using the ***d*** different hash functions.

```
mms[h_i(primary_key)] |= policy_string
```



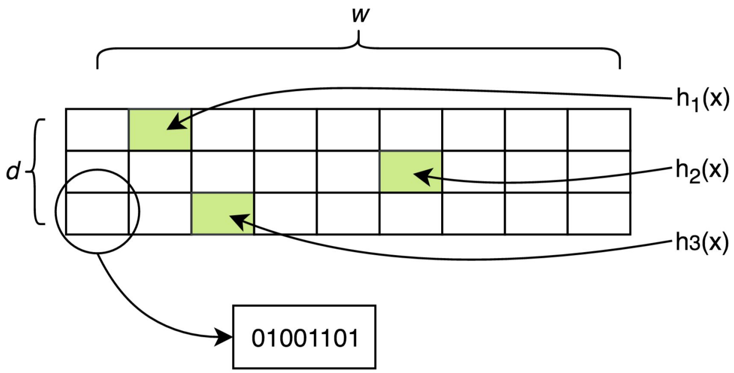
00101001

New element bit string

Min Mask Sketch: Inserting

- The new element is hashed based on its primary key (x) using the ***d*** different hash functions.

```
mms[h_i(primary_key)] |= policy_string
```

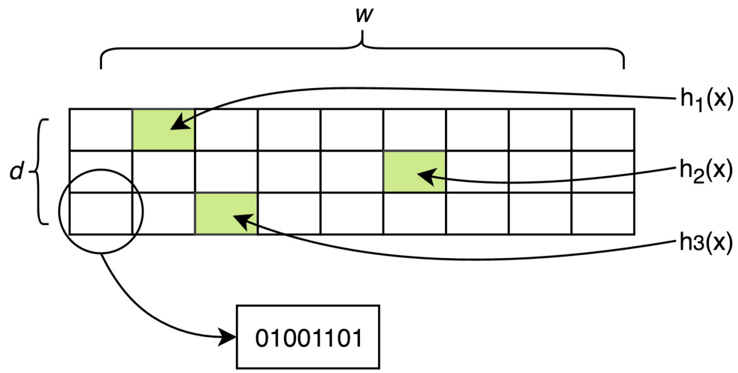


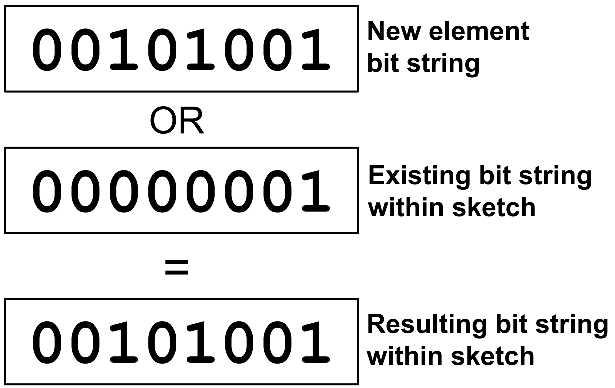
00101001	**New element bit string**
OR	
00000001	**Existing bit string within sketch**

Min Mask Sketch: Inserting

- The new element is hashed based on its primary key (x) using the ***d*** different hash functions.

```
mms[hi(primary_key)] |= policy_string
```





Min Mask Sketch: Querying

- An element is hashed based on its primary key (x) using the ***d*** different hash functions.

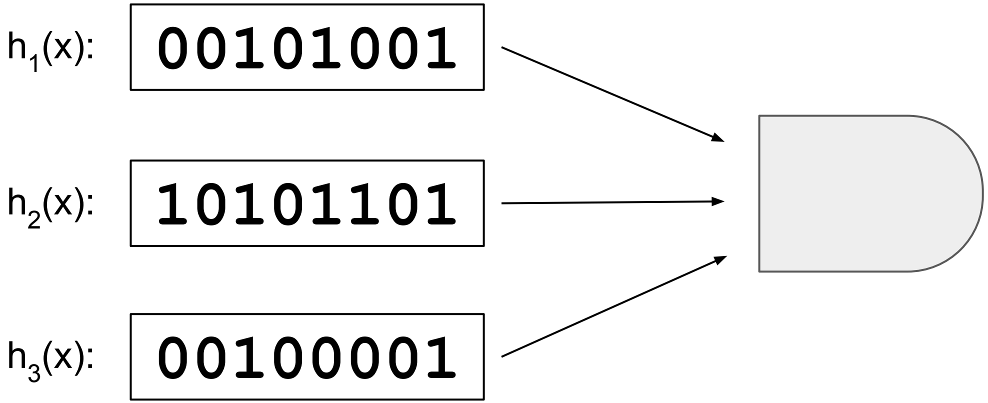
$h_1(x)$: `00101001`

$h_2(x)$: `10101101`

$h_3(x)$: `00100001`

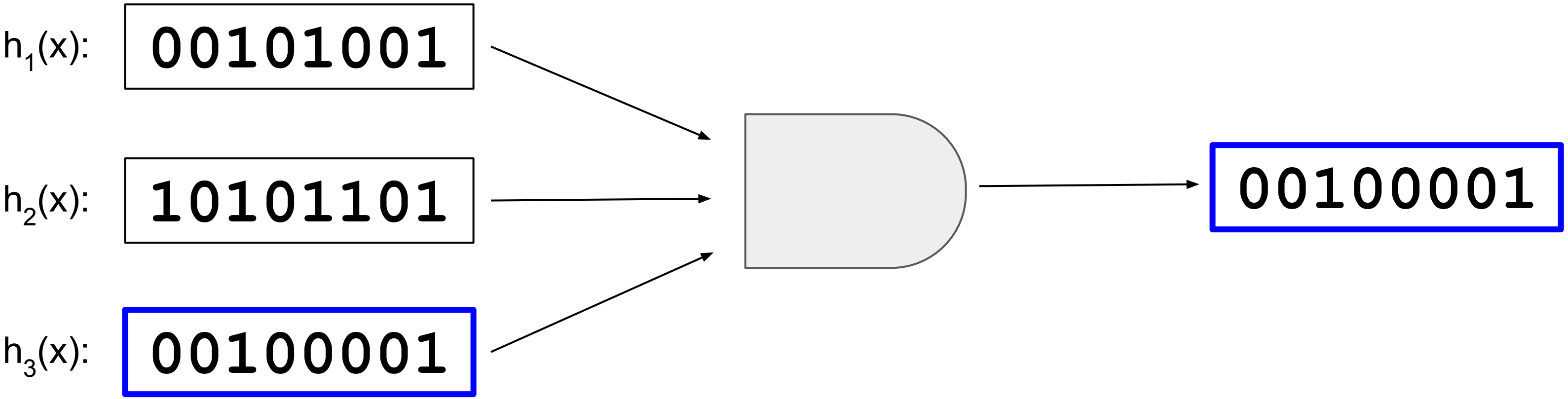
Min Mask Sketch: Querying

- An element is hashed based on its primary key (x) using the ***d*** different hash functions.
- The bit string with the minimum number of 1's (active expressions) is returned as the estimated sharing policy bit string.



Min Mask Sketch: Querying

- An element is hashed based on its primary key (x) using the ***d*** different hash functions.
- The bit string with the minimum number of 1's (active expressions) is returned as the estimated sharing policy bit string.



Implementation

- PostgreSQL version 9.6.
- Min Mask Sketch extension written in C.
- Extension contains the following components:

Implementation

- PostgreSQL version 9.6.
- Min Mask Sketch extension written in C.
- Extension contains the following components:
  - Definition of the Min Mask Sketch data type.

Implementation

- PostgreSQL version 9.6.
- Min Mask Sketch extension written in C.
- Extension contains the following components:
  - Definition of the Min Mask Sketch data type.
  - Functions to create a new Min Mask Sketch object.

Implementation

- PostgreSQL version 9.6.
- Min Mask Sketch extension written in C.
- Extension contains the following components:
  - Definition of the Min Mask Sketch data type.
  - Functions to create a new Min Mask Sketch object.
  - Functions to insert an element into the Min Mask Sketch.

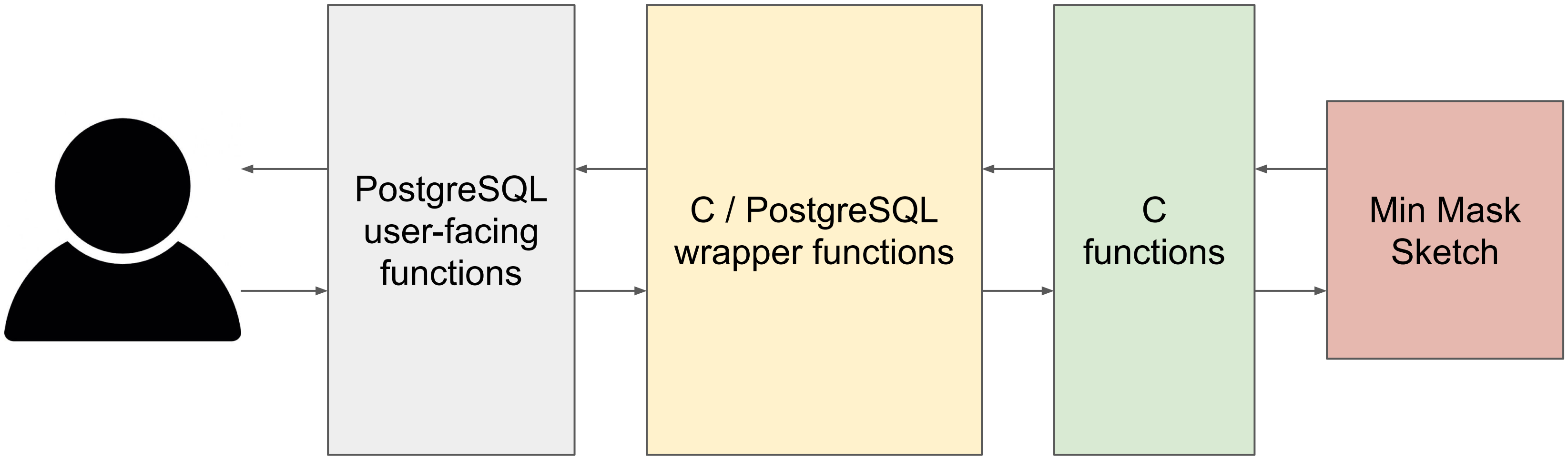
Implementation

- PostgreSQL version 9.6.
- Min Mask Sketch extension written in C.
- Extension contains the following components:
    - Definition of the Min Mask Sketch data type.
    - Functions to create a new Min Mask Sketch object.
    - Functions to insert an element into the Min Mask Sketch.
    - Functions to retrieve the bit string for a given element in the Min Mask Sketch.

Implementation

- PostgreSQL version 9.6.
- Min Mask Sketch extension written in C.
- Extension contains the following components:
  - Definition of the Min Mask Sketch data type.
  - Functions to create a new Min Mask Sketch object.
  - Functions to insert an element into the Min Mask Sketch.
  - Functions to retrieve the bit string for a given element in the Min Mask Sketch.
- https://github.com/oudalab/mms

Workflow


PostgreSQL
user-facing
functions
C / PostgreSQL
wrapper functions
C
functions
Min Mask
Sketch

Usage: Creating an Empty Min Mask Sketch

```
CREATE EXTENSION mms;

CREATE TABLE example (
    example_sketch mms
);

INSERT INTO example VALUES(mms());
```

Usage: Inserting an Element

```
UPDATE example SET example_sketch =
    mms_add(example_sketch, "abc"::text, 6);
```

Element Primary Key

00000110

Usage: Querying the Min Mask Sketch

```
SELECT mms_get_mask(example_sketch, "abc"::text)
    FROM example;
```

Benefit

- Consider the Health Tracker Pro example:

Benefit

- Consider the Health Tracker Pro example:
  - Each record takes 16 bytes to store.

time	heart_rate	blood_sugar	body_temp
2016-02-20 04:05:06	71	95	98.6

Benefit

- Consider the Health Tracker Pro example:
  - Each record takes 16 bytes to store.

time	heart_rate	blood_sugar	body_temp
2016-02-20 04:05:06	71	95	98.6

  - The simple approach of using 3 separate columns to store the sharing policy metadata would add an additional 3 bytes to each record.

Benefit

- Consider the Health Tracker Pro example:
  - Each record takes 16 bytes to store.

time	heart_rate	blood_sugar	body_temp
2016-02-20 04:05:06	71	95	98.6

  - The simple approach of using 3 separate columns to store the sharing policy metadata would add an additional 3 bytes to each record.
  - Using **c** = 95% and **ϵ** = 0.001, the Min Mask Sketch would require **8.154** KB to store the policy metadata.

Benefit

- Consider the Health Tracker Pro example:
  - Each record takes 16 bytes to store.

time	heart_rate	blood_sugar	body_temp
2016-02-20 04:05:06	71	95	98.6

  - The simple approach of using 3 separate columns to store the sharing policy metadata would add an additional 3 bytes to each record.
  - Using **c** = 95% and **ϵ** = 0.001, the Min Mask Sketch would require **8.154** KB to store the policy metadata.
  - For **1** GB of data, The simple approach would require **187.5** MB.

Benefit

- Consider the Health Tracker Pro example:
    - Each record takes 16 bytes to store.

time	heart_rate	blood_sugar	body_temp
2016-02-20 04:05:06	71	95	98.6

    - The simple approach of using 3 separate columns to store the sharing policy metadata would add an additional 3 bytes to each record.
    - Using **c** = 95% and **ϵ** = 0.001, the Min Mask Sketch would require **8.154** KB to store the policy metadata.
    - For **1** GB of data, The simple approach would require **187.5** MB.
    - This results in the Min Mask Sketch providing a **187.49** MB reduction in storage cost for this example.

Downside

- Could over-share data due to the probabilistic nature of the data structure.

Downside

- Could over-share data due to the probabilistic nature of the data structure.
- Cannot deactivate an expression (move from a 1 to a 0).

Downside

- Could over-share data due to the probabilistic nature of the data structure.
- Cannot deactivate an expression (move from a 1 to a 0).
- When policies cluster together, the mms can become inefficient.

Future Directions

- Expanding the Min Mask Sketch to store types of metadata other than sharing policy information.
- Rigorous study of the performance characteristics of the Min Mask Sketch.
- Comparison with other solutions to handling sharing policies.

References

Bloom, Burton H. "Space/time trade-offs in hash coding with allowable errors." *Communications of the ACM* 13.7 (1970): 422-426.

Cormode, Graham, and Shan Muthukrishnan. "An improved data stream summary: the count-min sketch and its applications." *Journal of Algorithms* 55.1 (2005): 58-75.

Kirsch, Adam, and Michael D. Mitzenmacher. "Building a better bloom filter." (2005).

Images Used

- http://cliparting.com/wp-content/uploads/2016/10/Young-person-clipart-kid.gif
- https://maxcdn.icons8.com/Share/icon/Data//database1600.png
- http://cliparting.com/wp-content/uploads/2017/01/Free-clip-art-doctor-clipartfest.jpeg
- https://upload.wikimedia.org/wikipedia/commons/thumb/3/36/Two_red_dice_01.svg/2000px-Two_red_dice_01.svg.png
- https://en.wikipedia.org/wiki/Bloom_filter#/media/File:Bloom_filter.svg
- https://i.stack.imgur.com/uh3NR.png
- https://raw.githubusercontent.com/docker-library/docs/01c12653951b2fe592c1f93a13b4e289ada0e3a1/postgres/logo.png

Thank You!

Policy Log Approach

- What if the data sharing policies tend to cluster together?

Policy Log Approach

- What if the data sharing policies tend to cluster together?

time	heart_rate	blood_sugar	body_temp	high_hr	low_bs	hide_bt
2016-02-20 04:05:06	71	95	98.6	0	1	0
2016-02-20 04:05:09	72	96	98.7	0	1	0
2016-02-20 04:05:09	72	94	98.7	0	1	0

⋮

2016-02-21 11:14:40	115	125	99.3	1	0	1
2016-02-21 11:14:43	115	124	99.5	1	0	1
2016-02-21 11:14:46	116	124	99.6	1	0	1

Policy Log Approach

- A log of the data sharing policies and when they change would be a better approach.
- This approach requires more space as a function of the policy changes.

key	high_hr	low_bs	high_bt
2016-02-20 04:05:06	0	1	0
2016-02-21 11:14:40	1	0	1

Min Mask Sketch vs. Policy Log

- In the context of the Health Tracker Pro example.
- Min Mask Sketch parameters:
  - **ϵ** = 0.001
  - **c** = 99%