Data Profiling

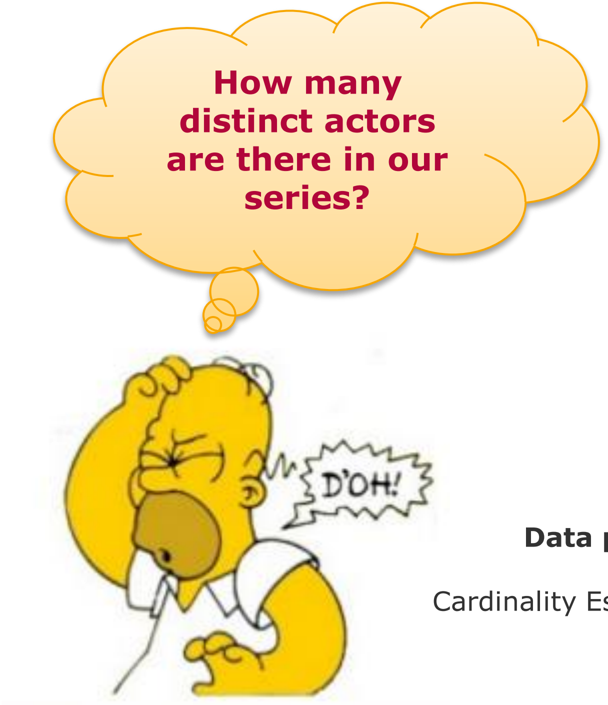
Cardinality Estimation

Hazar Harmouch

G-3.1.09, Campus III

Hasso Plattner Institut

Finding Number of Distinct Values Problem



Finding Number of Distinct Values Problem
A Polyonymous Problem

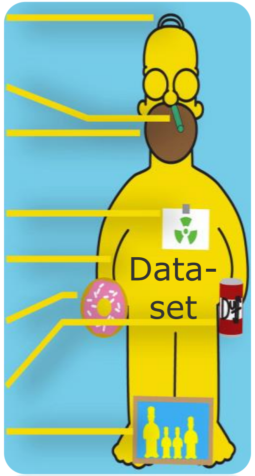
- Estimating the number of species in a population in statistics.
- „COUNT DISTINCT" and cardinality in DB literature.
- The zeroth-frequency moment of a multiset [Alon96].

The Zeroth-Frequency Moment of Dataset [Alon96]

- Let $E = (e_1, e_2, \dots e_n)$ be a multiset (e.g. Column in table or data stream) where each e_i is a member of a universe of N possible values and multiple items may have the same value.
- Let m_i denote the number of occurrences of i in the multiset E.
- The ***frequency moment*** of the multiset E is defined for each $k \geq 0$ as:

$$F_k = \sum_{i=1}^{n} m_i^{\,k}$$

  - F_0 is the number of distinct elements appearing in E or the cardinality of E.



Overview:
Cardinality Estimation

- **Motivation.**
- Cardinality estimation approaches.
- Cardinality estimation algorithms.
- Evaluation.

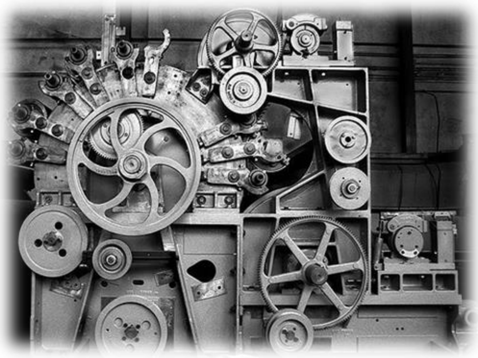
Cardinality Estimation Motivation
DBMS

- Database

Select … From… where…

- Query

- Statistics



- Query Optimizer

- Execution Plan

Cardinality Estimation Motivation
How Many Distinct …

… queries did I get?

…pairs (sourceIP,destinationIP) have I seen?

…distinct messages have I seen?

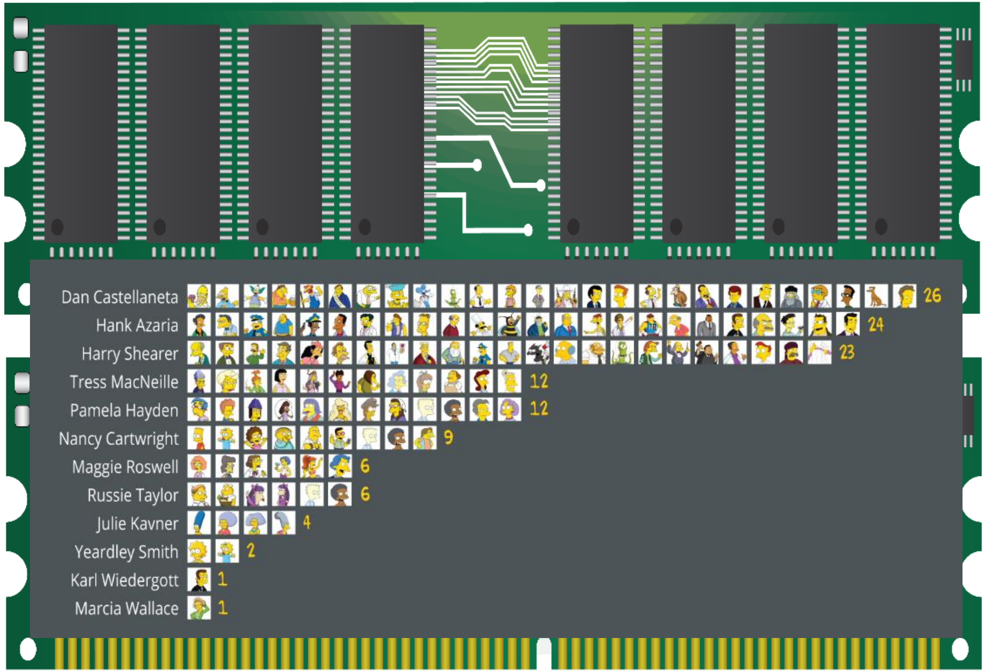
…values have I seen for this attribute x?

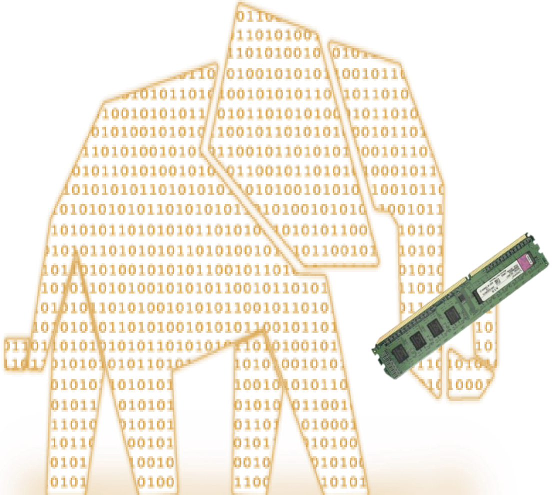
Is there any worm or denial of Service?

.. visitors to this website in order to advertise in it?

Cardinality Estimation Motivation

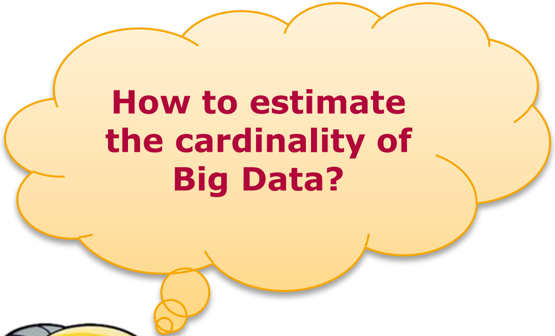
Big Data: Exact Counting is Not Easy! Estimate!





Exact cardinality of multiset determined with storage proportional to dataset size

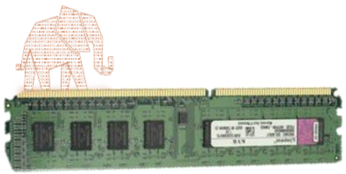
Cardinality Estimation
Big Data-Scale!



- Scale-up the computation
  - Expensive (hardware, equipment, energy).
  - Not always fast.

Or/and

- Scale-down the data
  - Create synopsis: data structure maintained by the estimation algorithm in main memory.
  - Need to fit the problem.

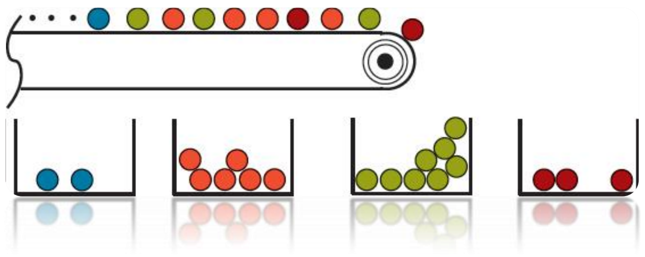


Overview:
Cardinality Estimation

- Motivation.
- **Cardinality estimation approaches.**
- Cardinality estimation algorithms.
- Evaluation.

Exact cardinality: Sorting

- Sorting eliminates duplicates.



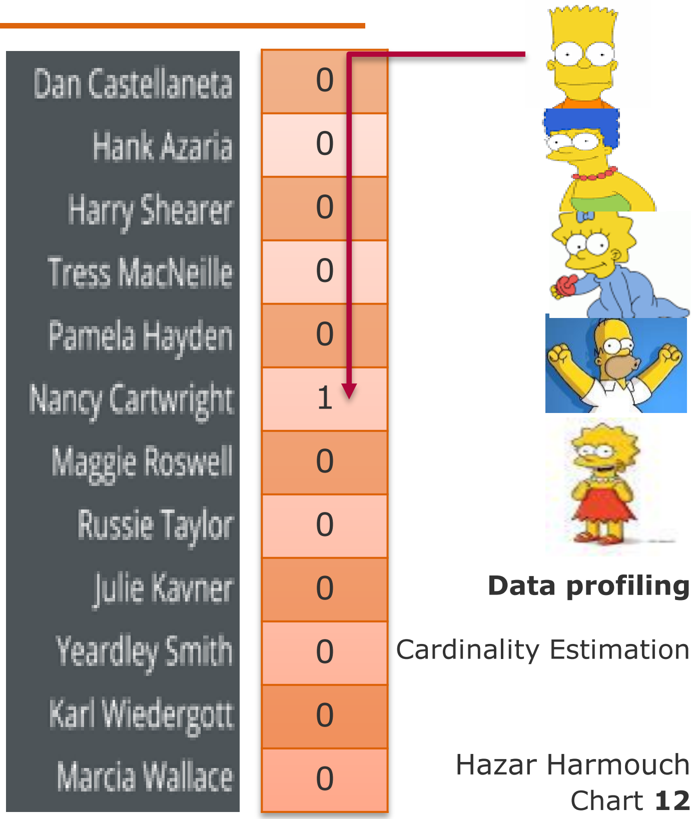
- **Problem**:
  - Expensive operation.
  - Synopsis size is at least as large as the dataset itself.
  - Impractical for current big datasets.

Exact Cardinality: Bitmap

- **Synopsis**: is a bitmap of size equals to universe size and initialized to 0s.
  - Scan dataset once and set the bit i to 1 whenever an item with the i –th value of the universe is observed.
  - Cardinality= Number of 1s.

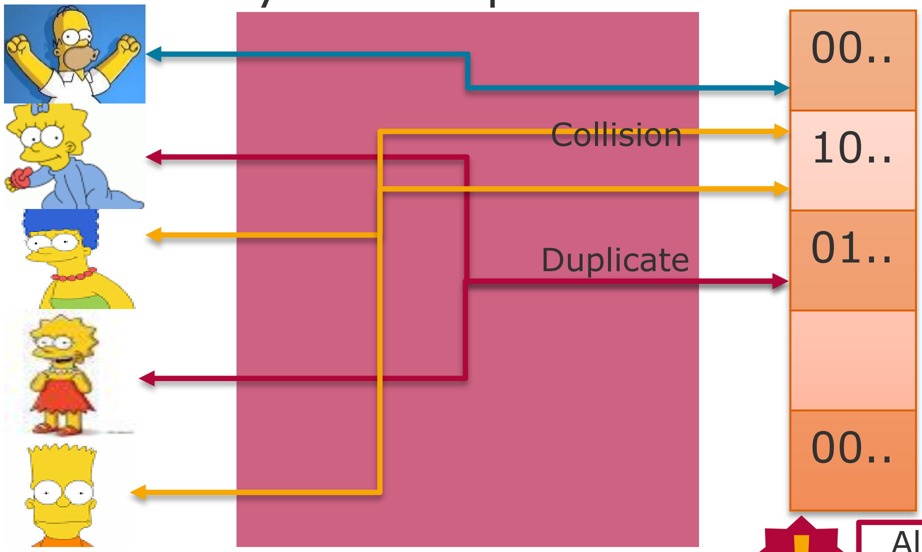
- **Problem**: The synopsis size is a function of the universe size N, which is potentially much larger than the size of the dataset itself.

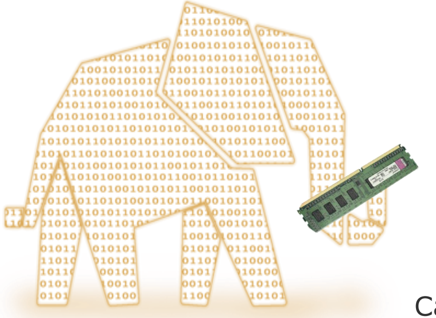
! Still used in another approached.



Cardinality Estimation Approaches
Exact Cardinality: Hashing

- Hashing eliminates duplicates without sorting and requires one pass.
- Simple application of hashing can be worse than sorting in terms of memory consumption.

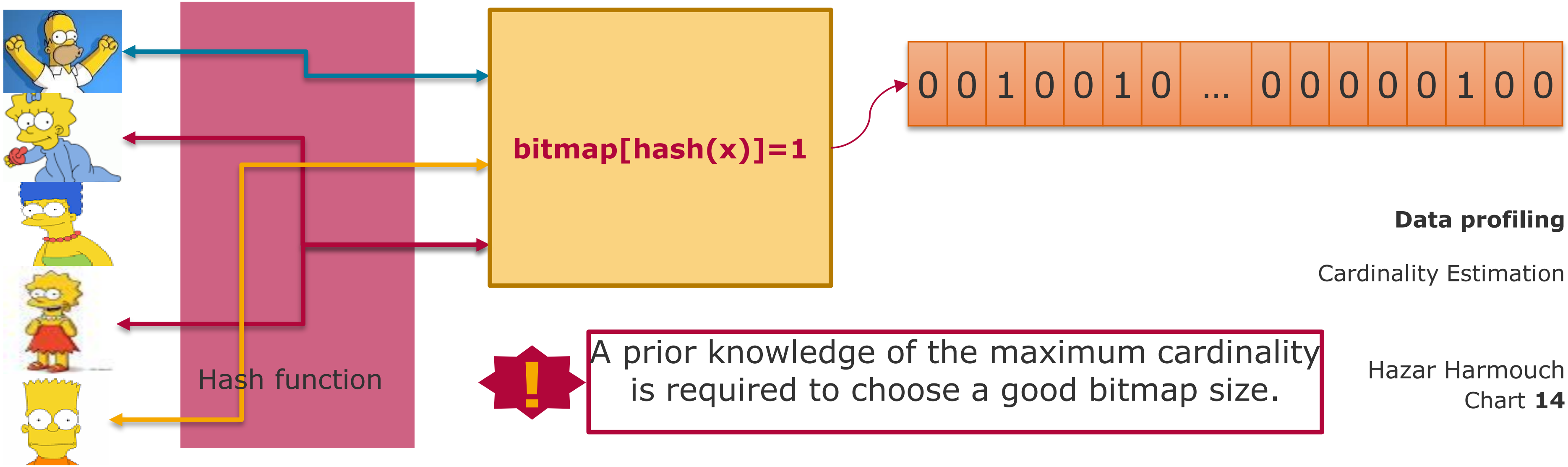




All exact approaches are expensive in both size and runtime.

Estimation: Bitmap of hash values

- Scales down the synopsis size by don't store the hash values.
- **Synopsis**: a bitmap keeps track of the hashed values.
  - The hash function maps each item to a bit in the bitmap.
  - Like Bloom Filters.



A prior knowledge of the maximum cardinality is required to choose a good bitmap size.

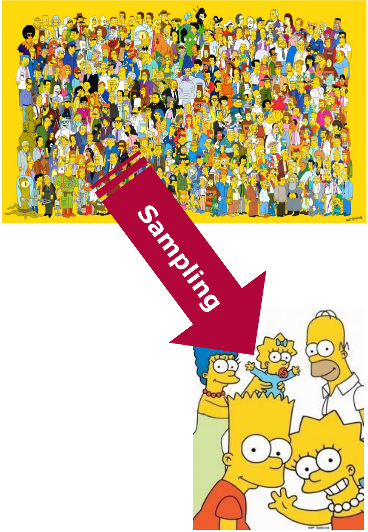
Cardinality Estimation approaches

Estimation: Sampling

- Several negative results.

- For every estimate based on a small-sample, there is a dataset where the ratio error can be made arbitrarily large [Charikar00].

- Almost all the dataset needs to be sampled to bound the estimation error within a small constant [Haas95, Haas98].

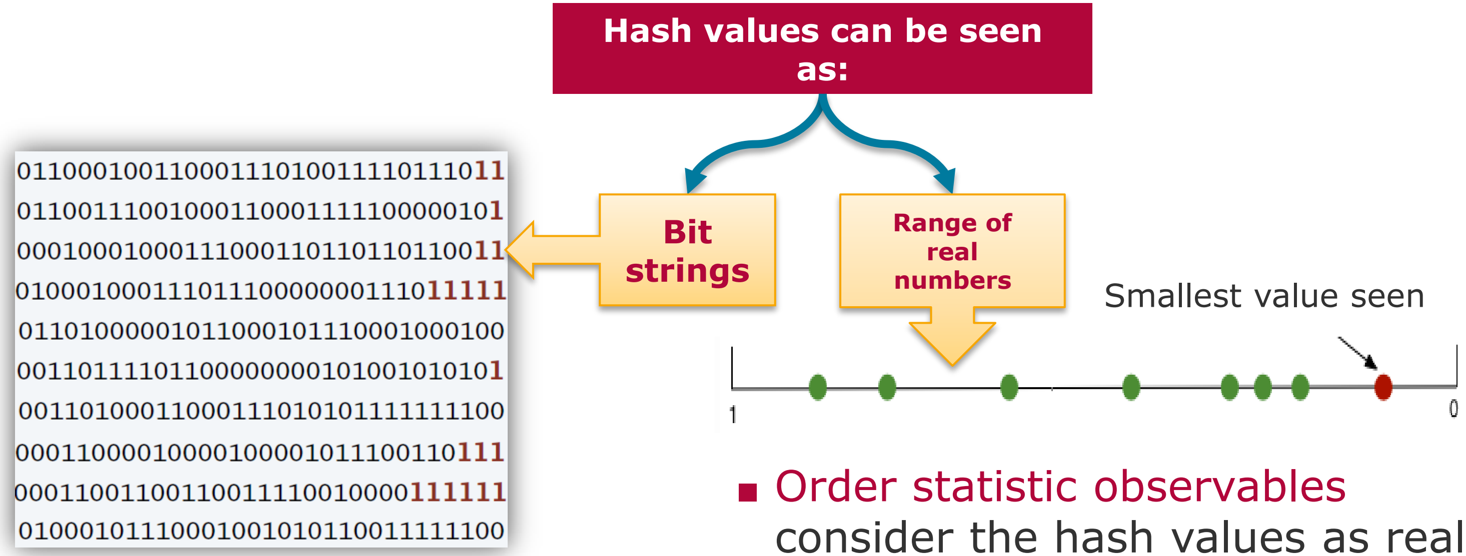


Estimation: Observations in hash values (1/2)

- Find a suitable observation on the hashed values:
  - Depends only on the input multiset.
  - Independent of replications.
  - Maintained with few registers.
- Inferring a estimate of the unknown cardinality.

- Two Categories:
  - Bit pattern observables
  - Order statistics observables

Estimation: Observations in hash values (2/2)



- Bit pattern observables depends on the occurrence of particular bit patterns at the binary string representation.
- Order statistic observables consider the hash values as real numbers.
- The order statistic of rank k is the k-th smallest value in the dataset.

Overview:
Cardinality Estimation

- Motivation.
- Cardinality estimation approaches.
- **Cardinality estimation algorithms**
- Evaluation.

Algorithms

Algorithm	Year	Ref.
Flajolet-Martin **(FM)**	1985	[Flajolet-Martin85]
Probabilistic Counting with Stochastic Averaging **(PCSA)**	1985	[Flajolet-Martin85]
Alon, Martias, and Szegedy Algorithm **(AMS)**	1996	[Alon96]
Ziv Bar-yossef , T. S. Jayram , Ravi Kumar , D. Sivakumar , Luca Trevisan Algorithm **(BJKST)**	2002	[Bar02]
LogLog	2003	[Durand-Flajolet03]
SuperLogLog	2003	[Durand-Flajolet03]
HyperLogLog	2008	[Flajolet08]
Hyperloglog++	2013	[Heule13]
MinCount	2005	[Giroire09]
AKMV	2007	[Beyer07]
LC	1990	[Wang90]
BF	2010	[Papapetrou10]

Cardinality Estimation
Algorithms Classification

Algorithm	Observables	Intuition	Core method
FM	Bit-pattern	Logarithmic hashing	Count trailing 1s
PCSA	Bit-pattern	Logarithmic hashing	Count trailing 1s
AMS	Bit-pattern	Logarithmic hashing	Count leading 0s
BJKST	Order statistics	Bucket-based	Count leading 0s
LogLog	Bit-pattern	Logarithmic hashing	Count leading 0s
SuperLogLog	Bit-pattern	Logarithmic hashing	Count leading 0s
HyperLogLog	Bit-pattern (order statistics)	Logarithmic hashing	Count leading 0s
HyperLogLog++	Bit-pattern	Logarithmic hashing	Count leading 0s
MinCount	Order statistics	Interval-based	k-th minimum value
AKMV	Order statistics	Interval-based	k-th minimum value
LC	No observable	Bucket-based	Linear synopses
BF	No observable	Bucket-based	Linear synopses

Flajolet-Martin (FM) [Flajolet-Martin85]

As I said over the phone, I started working on your algorithm when Kyu-Young Whang considered implementing it and wanted explanations/estimations. I find it simple, eleg
and amazingly powerful.

Without analysis (original algorithm)

```
After all the values have been processed, then
if M(MAP)=000,  then RESULT=LO(MAP)-1
if M(MAP)=111,  then RESULT=LO(MAP)+1
otherwise RESULT=LO(MAP).

For example,
   if MAP was 0000000000000000000000001111111
   LO(MAP) is 8 and M(MAP) is 000: RESULT=7
   if MAP was 00000000000000000000011101111111
   LO(MAP) is 8 and M(MAP) is 111: RESULT=9
   if MAP was 00000000000000000000001001111111
   LO(MAP) is 8 and M(MAP) is 010: RESULT=8
```

With analysis (Philippe)

Philippe determines that

$$\mathbb{E}[2^P] \approx \phi n$$

where $\phi \approx 0.77351\ldots$ is defined by

$$\phi = \frac{e^\gamma\sqrt{2}}{3}\prod_{p=1}^{\infty}\left[\frac{(4p+1)(4p+2)}{(4p)(4p+3)}\right]^{(-1)^{\nu(p)}}$$

such that we can apply a simple correction and have unbiased estimator,

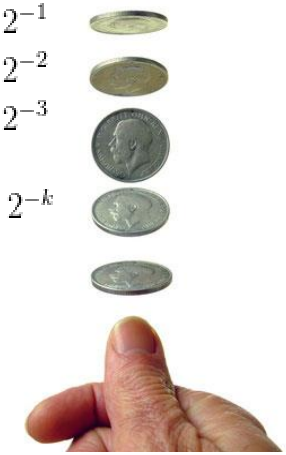
$$Z := \frac{1}{\phi}2^P \qquad \mathbb{E}[Z] = n$$

Photos:https://speakerdeck.com/timonk/philippe-flajolets-contribution-to-streaming-algorithms

Flajolet Martin (FM) [Flajolet-Martin85]

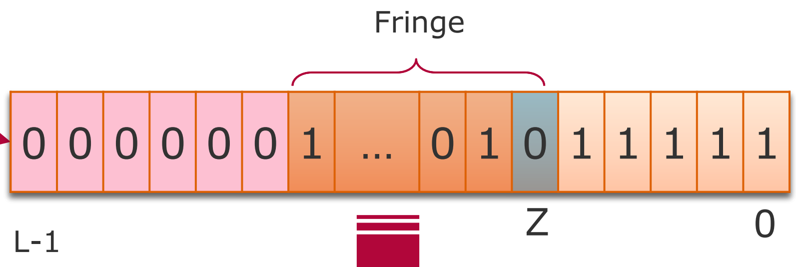
- First algorithm for cardinality estimation (1983).
- Experimental observation:
  - If the hash values are uniformly distributed, the prefix $..10^k$ appears with probability $1/2^{(k+1)}$.
- The observable $\rho\big(hash(x_i)\big)$ is the position of the least significant 1-bit in $hash(x_i)$.
  - ρ(0011001100110110)=1
  - All stings with same ρ represented once (no duplicate).
- Intuition:
  - Seeing $\rho = k$ means there are at least 2^{k+1} different bit strings.
  - Find the largest ρ and estimate the cardinality by 2^{ρ}.

$$
\begin{aligned}
P(.....1) &= 2^{-1} \\
P(....10) &= 2^{-2} \\
P(...100) &= 2^{-3} \\
&\vdots \\
P(...10^{k-1}) &= 2^{-k}
\end{aligned}
$$



Flajolet Martin (FM) [Flajolet-Martin85]

- **Bitmap[rho(hash(x))]=1**
- **rho(y)=position of the LSB=1 in y.**



Z:Number of trailing 1s in the bitmap

L: length of the hash bit string (e.g. 32 bit)

Estimate $F_0 = \lfloor 2^Z / 0.77351 \rfloor$

Flajolet Martin (FM) [Flajolet-Martin85]

Item value v	Hash Function ρ	$M[\cdot]$
15	1	00000010
36	0	00000011
4	0	00000011
29	0	00000011
9	3	00001011
36	0	00001011
14	1	00001011
4	0	00001011

$Z = 2$

Estimate $F_0 = \left\lfloor \frac{2^2}{0.77351} \right\rfloor = 5$

Probabilistic Counting (FM): Direct Averaging

Bitmap1[rho(hash1(x))]=1

Bitmap2[rho(hash2(x))]=1

:

Bitmapm[rho(hashm(x))]=1

L-1 ... 0

Bitmap1: 0 0 0 0 0 0 1 ... 0 1 0 1 1 1 1 1

Bitmap2: 0 0 0 0 0 0 1 ... 0 1 0 1 0 1 1 1

:

Bitmap$m$: 0 0 0 0 0 0 1 ... 0 1 0 0 1 1 1 1

Reduced Variance

m Hash functions

Standard Error$=1/\sqrt{m}$

Memory$= L * m$

Estimate $F_0 = \lfloor 2^{z_{average}} / 0.77351 \rfloor$

$$z_{average} = (z_1 + \cdots + z_m)/m$$

Cardinality Estimation
Flajolet Martin (FM) [Flajolet-Martin85]

- m: Number of hash functions.
- $B_1 \ldots B_m$: Bitmaps of length L initialized to 0s.
- $h_1 \ldots h_m$: uniform hash functions.
- $L > \log_2\left(\frac{N_{max}}{m}\right) + 4$ and N_{max} maximum cardinality to which we safely want to count up to.
- $\rho(x)$: The position of the least significant 1-bit in the bit string x.

- Determine m according to the desired standard error.
- Scan dataset once and for each item x:
  - Set $B_1[\rho(h_1(x))] = 1 \ldots B_m[\rho(h_m(x))] = 1$
- Find $z_1 \ldots z_m$ the position of the least significant 0 in the corresponding bitmap or number of trailing 1s.
- Estimate $F_0 = \lfloor 2^{Zaverage}/0.77351 \rfloor$

Cardinality Estimation
Flajolet Martin (FM) [Flajolet-Martin85]

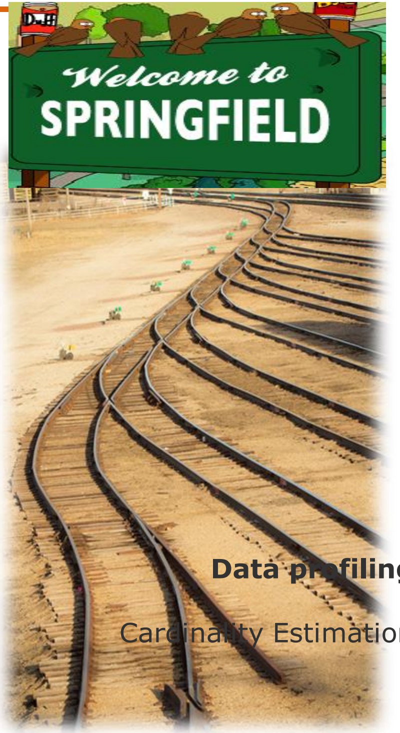
Item value v	Hash Function 1 ρ	Hash Function 1 $M[\cdot]$	Hash Function 2 ρ	Hash Function 2 $M[\cdot]$	Hash Function 3 ρ	Hash Function 3 $M[\cdot]$
15	1	00000010	1	00000010	0	00000001
36	0	00000011	1	00000010	0	00000001
4	0	00000011	0	00000011	0	00000001
29	0	00000011	2	00000111	1	00000011
9	3	00001011	0	00000111	0	00000011
36	0	00001011	1	00000111	0	00000011
14	1	00001011	0	00000111	1	00000011
4	0	00001011	0	00000111	0	00000011
		$Z=2$		$Z=3$		$Z=2$

$$\text{Estimate } \hat{F}_0 = \left\lfloor \frac{2^{(2+3+2)/3}}{.77351} \right\rfloor = 6$$

Cardinality Estimation

PCSA: Probabilistic Counting with Stochastic Averaging [Flajolet-Martin85]

- Same observable.
- Emulate the effect of m experiments with single hash function.
- m Bitmap: Each responsible of $\frac{F_0}{m}$ items of the dataset.
- Get m observables and then estimate using the average.



Example:

hash(x)= 0011001100110110

m=4

bitmap2[0]=1

L-1 ▪ Hash Value 0

0	0	0	1	0	0	1	…	1	0	0	0	1	1	0	1

$\mathrm{L}-\log_2(m)$bits Used to update the bitmap

$\log_2(m)$ bits Determine the bitmap

Cardinality Estimation

PCSA: Probabilistic Counting with Stochastic Averaging [Flajolet-Martin85]

i=hash(x)%m

Bitmapi[rho(hash(x)/m)]=1

$\mathrm{L}-\log_2(m)-1$ … 0

0	0	0	0	0	0	1	…	0	1	0	1	1	1	1	1

Bitmap1

0	0	0	0	0	0	1	…	0	1	0	1	0	1	1	1

Bitmap2

:

0	0	0	0	0	0	1	…	0	1	0	0	1	1	1	1

Bitmapm

Standard Error=$0.78/\sqrt{m}$

Memory= $(\mathrm{L}-\log_2(m)) * m$

Estimate $F_0 = \lfloor m * 2^{Zaverage}/0.77351 \rfloor$

PCSA: Probabilistic Counting with Stochastic Averaging [Flajolet-Martin85]

- m: Number of bitmaps.
- $B_1 \dots B_m$: Bitmaps of length L initialized to 0s.
- h: uniform hash function.
- $\rho(x)$: The position of the least significant 1-bit in the bit string x.

- Determine m according to the desired standard error.
- Scan dataset once and for each item x:
  - Set $B_i[\rho(h(x)/m)] = 1$ where $i = h(x)\%m$
  - Find $z_1 \dots z_m$ the position of the least significant 0 in the corresponding bitmap or number of trailing 1s.
- Estimate $F_0 = \lfloor m * 2^{Zaverage}/0.77351 \rfloor$

Cardinality Estimation
Alon, Martias, and Szegedy Algorithm (AMS) [Alon96]

Bitmap for FM

L-1 … R … Fringe … Z … 0

0	0	0	0	0	0	1	…	0	1	0	1	1	1	1	1

- Like FM, but:
  - Uses position of the most significant bit set to 1 (or the number of leading 0s) in the map for cardinality estimation.
  - Doesn't maintain a bitmap but keeps track only of the largest observable ρ.

Synapsis for AMS

Max_{m-1}	Max_{m-2}	Max_{m-3}	…	max_3	max_2	max_1	max_0

! m hash function

Single outlier can set R

R: Number of trailing 0s in bitmap

Estimate $F_0 = 2^{Raverage}$

LogLog Algorithm [Durand-Flajolet03]

- Reduce the synopsis size from $\log_2(n)$ to $\log_2(\log_2 n)$.
- PCSA like with AMS capacity reduction.

Log2{  }

i=hash(x)%m
Mi=max{Mi, rho(hash(x)/m)}

m-1							0
M_{m-1}	M_{m-2}	M_{m-3}	…	M_3	M_2	M_1	M_0

Estimate $F_0 = \alpha_m * m * 2^{Maverage}$

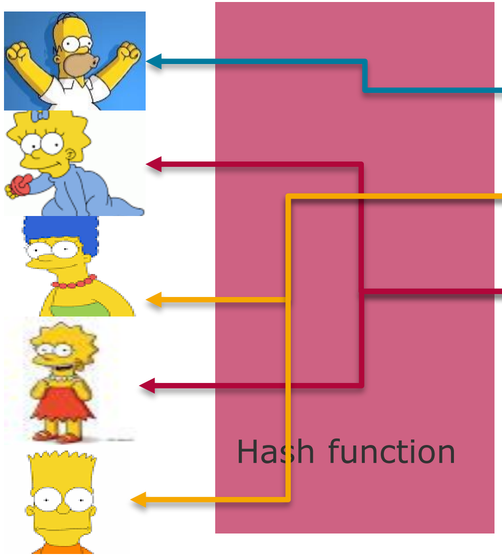
$\alpha_m = 0.39701 \; as \; soon \; as \; m \geq 64$

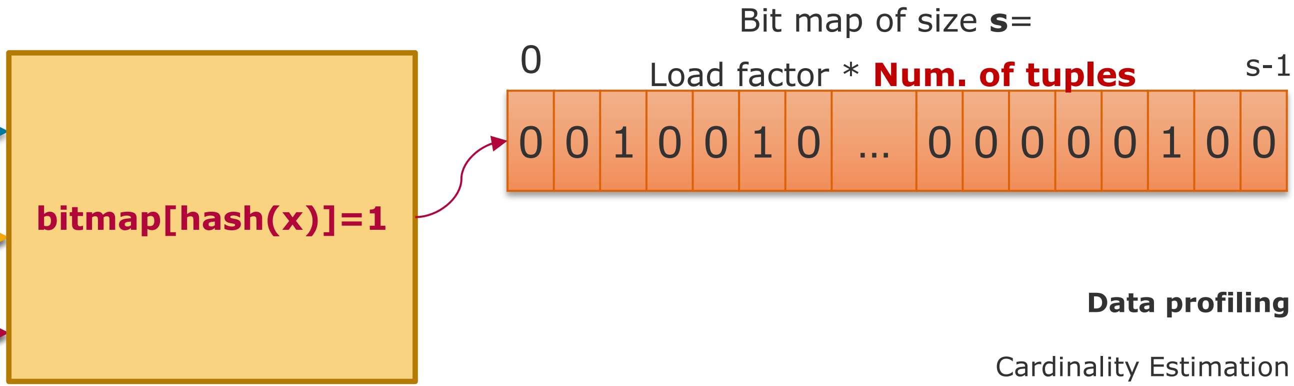
Solves AMS problem of using m hash functions

Still has AMS problem: Single outlier can set R

std. Error=$1.3/\sqrt{m}$

Linear Counting [Whang90]





Intuition:

✓ Number of 1s is not good estimate due to collisions.

✓ How empty is the bitmap?

Estimate $F_0 = -s * \ln(V)$

V is the number of 0s divided by bitmap size.

Cardinality Estimation Algorithms
Linear Counting [Whang90]

- Measure the dataset number of tuples as n_{max}.
- Specify the desired accuracy (i.e. standard error) and calculate the load factor t using:

$$std.err. = \sqrt{\frac{(e^t - t - 1)}{(t * n_{max})}} = 0.01\ for\ t \geq 12$$

!

Memory consumption grows linearly as a function of the expected cardinality which can be as large as dataset size.

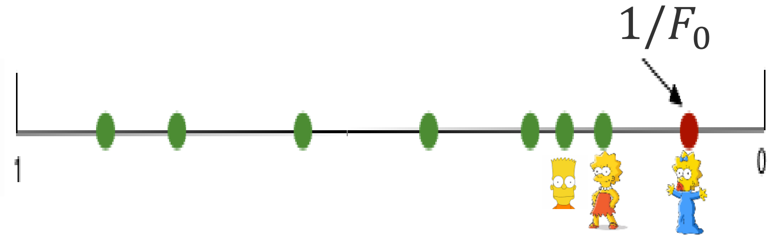
- Allocate a bitmap B of size $s = n_{max} * t$ and initialize it to 0s.
- Use a uniform hash function h.
- Scan the datasets once and set $B[h(x_i)] = 1$ for each item x_i.
- Cardinality estimated by $F_0 = -s * \ln V$
  - where V is the number of bits who still 0s divided by s.

AKMV Algorithm [Beyer07]

- Hash values are evenly distributed over the range.
- Estimate the number of distinct values you have seen by knowing the average spacing between values in the range.
  - If we have F_0distinct values, we expect them on average to be spaced about $1/F_0 th$ apart from each other.
  - So, the k-th minimum value is a good estimate of k/F_0

Estimate $F_0 = (k-1)/M^{(k)}$

$M^{(k)}$ is the k-th smallest value seen.



Overview:
Cardinality Estimation

- Motivation.
- Cardinality estimation approaches.
- Cardinality estimation algorithms.
- **Evaluation.**

Evaluation

- Some applications require a very accurate estimation. However, others accept a less accurate estimation.
  - The number of distinct visitors of a website influences the price of showing advertisements. So allowing only a small error in measuring the cardinality is important.
  - A good estimation of the number of distinct connections is enough to detect a potential denial of service attack.

- Evaluation metrics:
  - Accuracy
  - Runtime.

Evaluation

- **Implementations** (Unified test environment):
  - Implemented for Metanome.
  - MurmurHash.
  - all algorithms were configured to produce theoretical (standard/relative) errors of 1%.
- **Datasets**: 90 synthetic datasets. The exact cardinalities were made to be the powers of 10, starting with 10 up to 10^9.

Dataset	# Attributes	# Tuples
NCVoter	25 (of 71)	7,560,886
Openadresses-Europe	11	93,849,474

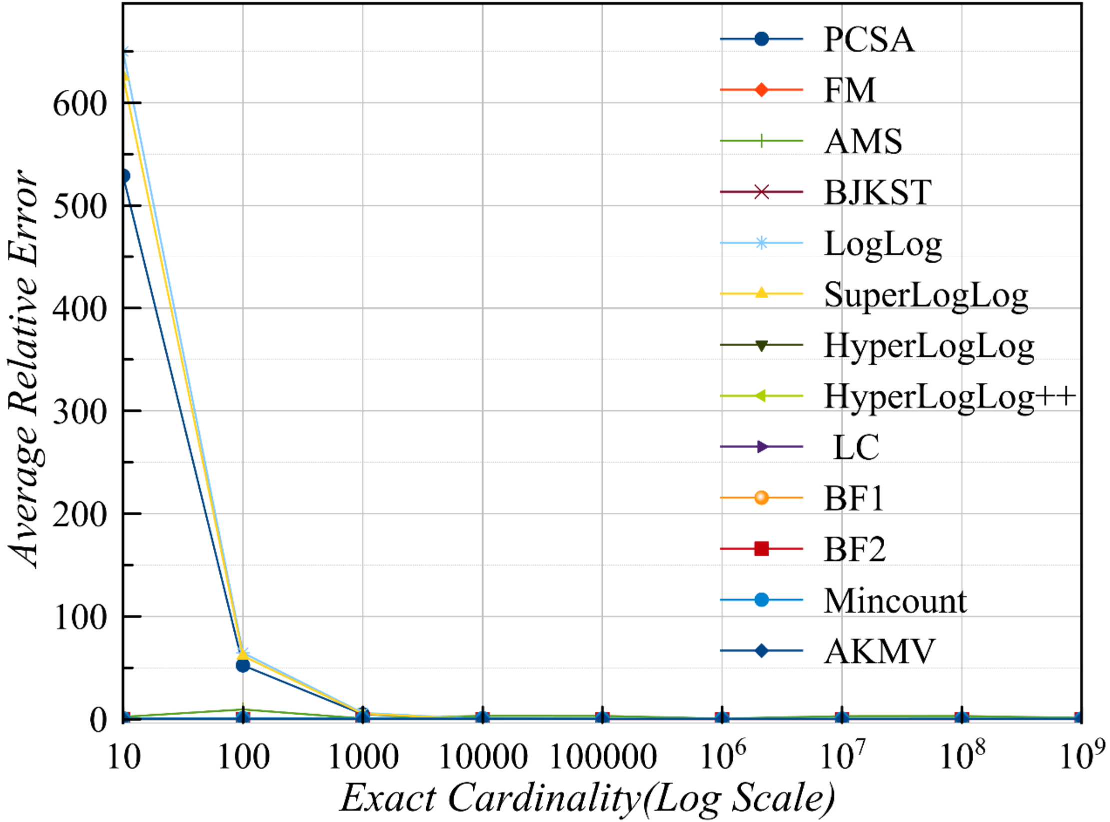
Evaluation

- **Evaluation metrics:**
  - Same memory capacity.
  - Evaluate their performance regarding runtime and estimation accuracy.

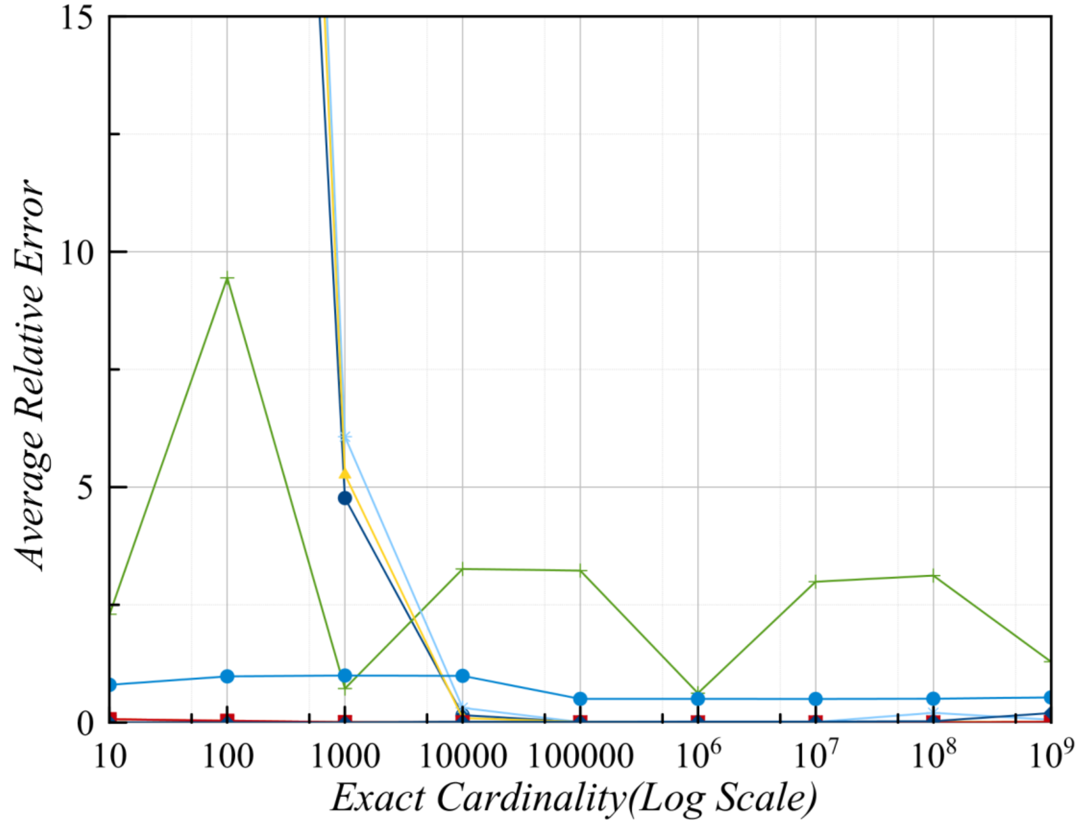
- **Runtime**: total time taken by an algorithm to process all the data elements and estimate the cardinality.

- **Accuracy**: relative error.

Cardinality Estimation
Evaluation-Accuracy



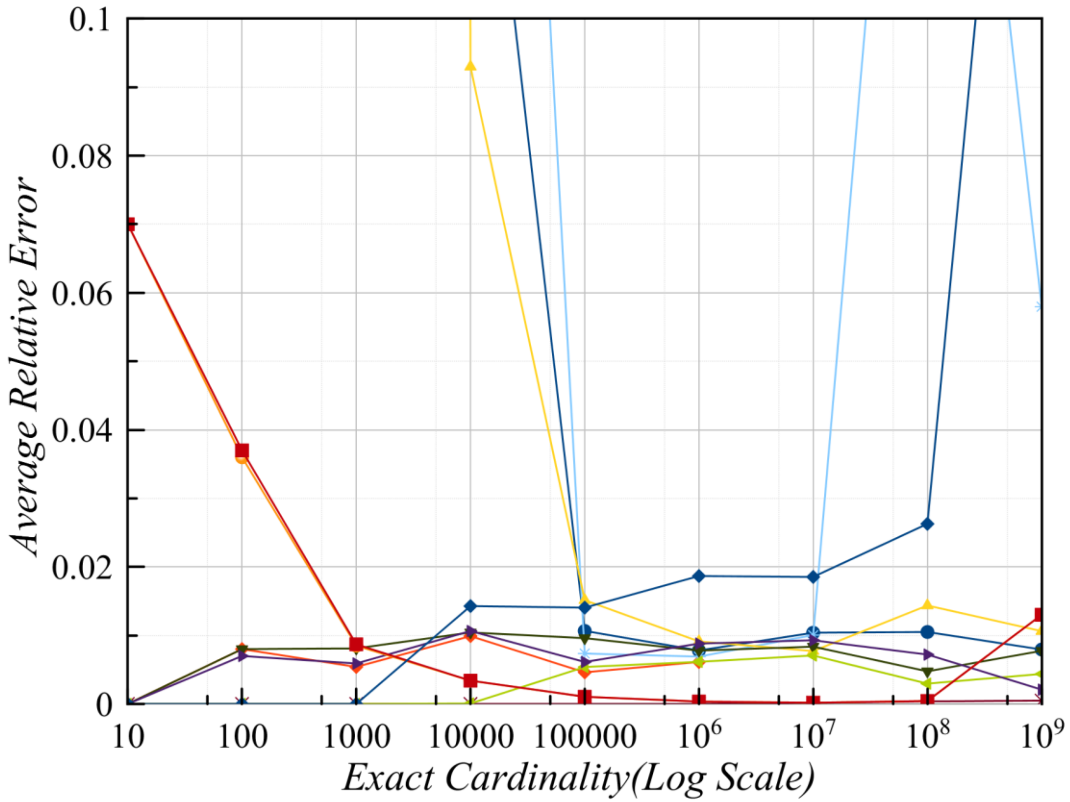
Cardinality Estimation
Evaluation-Accuracy



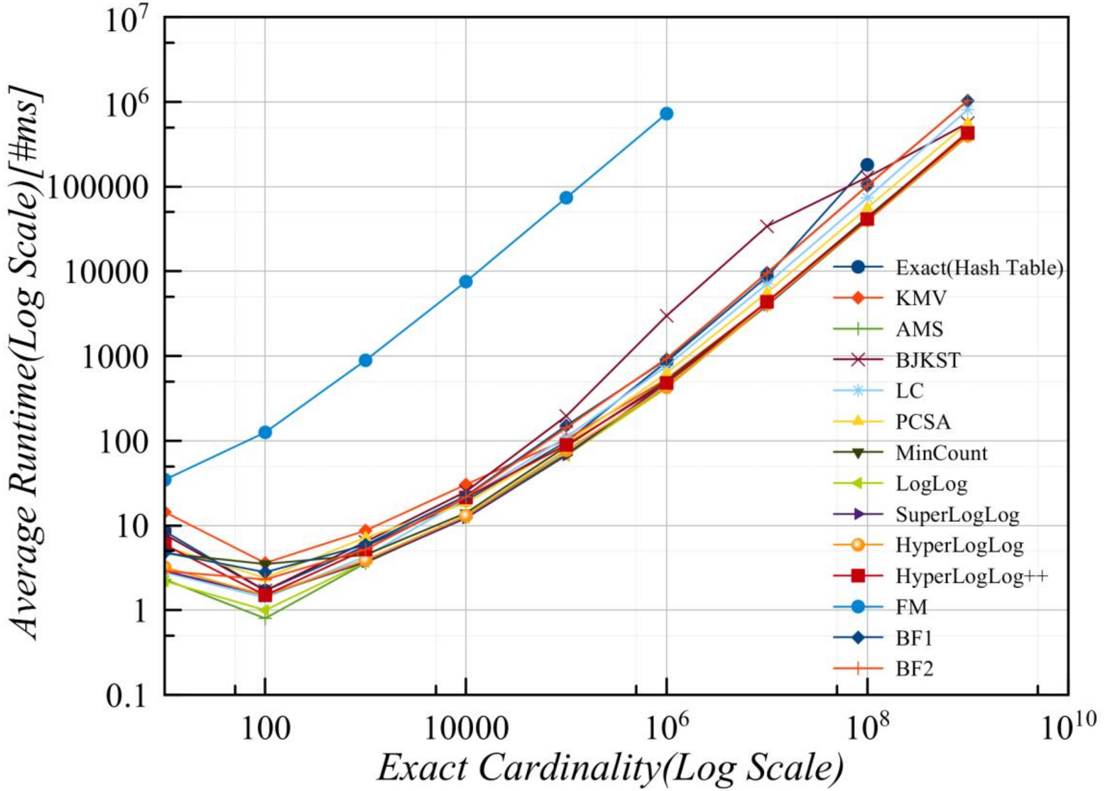
Data profiling

Cardinality Estimation

Cardinality Estimation
Evaluation-Accuracy

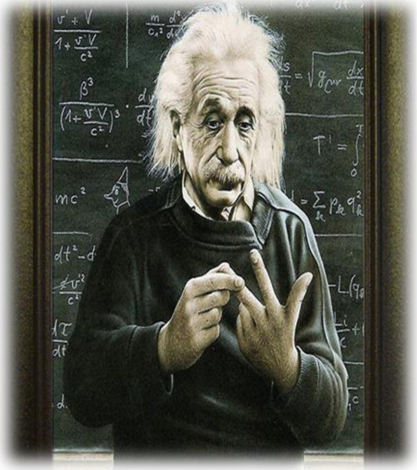


Cardinality Estimation
Evaluation- Run Time



Summary

- Cardinality of dataset has several application in many domains.
- Exact cardinality approaches: Sorting, bitmap, and hashing
- Exact approaches are expensive in both size and runtime.
- Estimation approaches: bitmap of hashes, sampling, and observations in hash values.
- Estimation algorithm families.
  - FM, PCSA: Count trailing 1s.
  - AMS, LogLog: Count leading 0s.
  - LC: linear synopsis.
  - AKMV: k-minimum hash values.
- Evaluation is good to choose the right algorithm for your application.



References

- [Charikar2000] M. Charikar, S. Chaudhuri, R. Motwani, and V. Narasayya. Towards Estimation Error Guarantees for Distinct Values. In Proceedings of the 19th ACM PODS Symposium on Principles of Database Systems, pages 268–279, 2000.
- [Haas98] P. Haas and L. Stokes. Estimating the Number of Classes in a Finite Population. Journal of the American Statistical Association, 93:1475–1487, 1998.
- [Haas95] Haas, P. J., Naughton, J. F., Seshadri, S., & Stokes, L. (1995, September). Sampling-based estimation of the number of distinct values of an attribute. In *VLDB* (Vol. 95, pp. 311-322).
- [Bar02] Bar-Yossef, Ziv, et al. Counting distinct elements in a data stream.Randomization and Approximation Techniques in Computer Science. Springer Berlin Heidelberg, 2002. 1-10.
- [Heule13] Heule, S., Nunkesser, M., & Hall, A. (2013). HyperLogLog in practice: algorithmic engineering of a state of the art cardinality estimation algorithm. In Proceedings of the 16th International Conference on Extending Database Technology.

References

- [Wang90] Whang, K. Y., Vander-Zanden, B. T., & Taylor, H. M. (1990). A linear-time probabilistic counting algorithm for database applications. *ACM Transactions on Database Systems (TODS)*, *15*(2), 208-229.
- [Flajolet-Martin85] Flajolet, P., & Martin, G. N. (1985). Probabilistic counting algorithms for data base applications. *Journal of computer and system sciences*, *31*(2), 182-209.
- [Alon96] Alon, N., Matias, Y., & Szegedy, M. (1996, July). The space complexity of approximating the frequency moments. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing* (pp. 20-29). ACM
- [Durand-Flajolet03] Durand, M., & Flajolet, P. (2003). Loglog counting of large cardinalities. In*Algorithms-ESA 2003* (pp. 605-617). Springer Berlin Heidelberg.
- [Flajolet08] Flajolet, P., Fusy, É., Gandouet, O., & Meunier, F. (2008). Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. *DMTCS Proceedings*, (1).

References

- [Giroire09] *F. Giroire. Order statistics and estimating cardinalities of massive data sets. Discrete Applied Mathematics, 157(2):406{427, 2009.*
- [Papapetrou10] *O. Papapetrou, W. Siberski, and W. Nejdl. Cardinality estimation and dynamic length adaptation for Bloom filters. Distributed and Parallel Databases, 28(2):119-156, 2010.*
- [Beyer07] *K. Beyer, P. J. Haas, B. Reinwald, Y. Sismanis, and R. Gemulla. On synopses for distinct-value estimation under multiset operations. In Proceedings of the International Conference on Management of Data (SIGMOD), pages 199-210, 2007.*