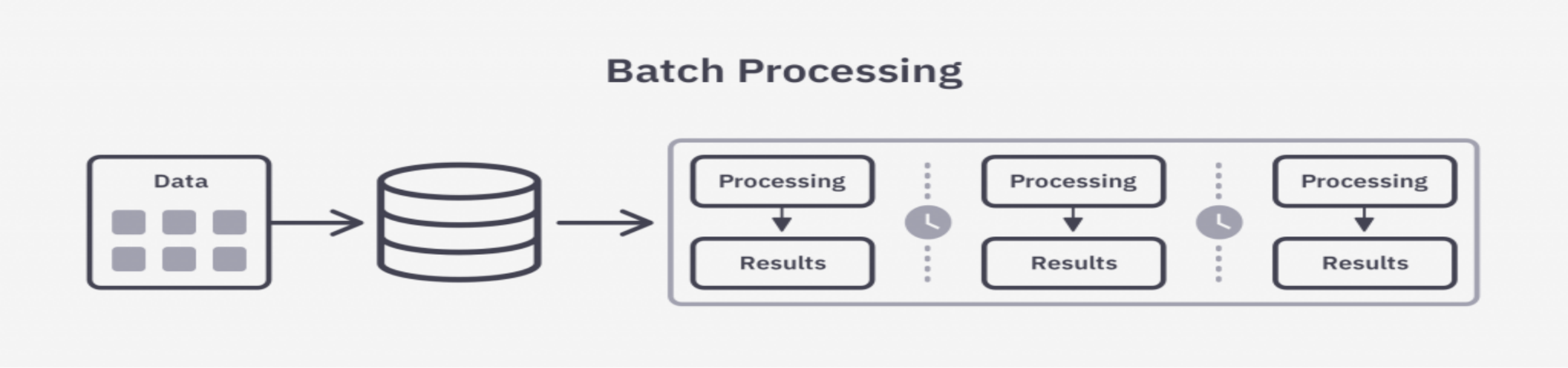
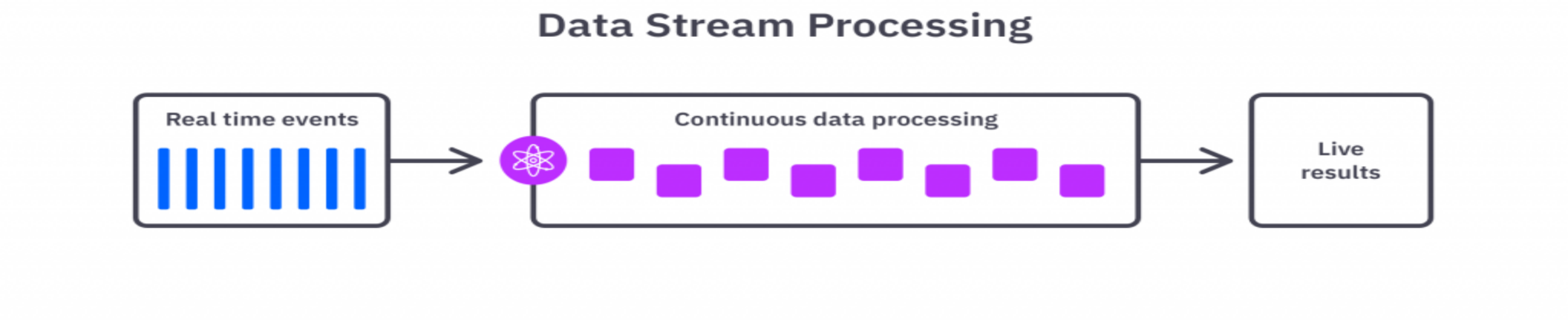
Data Streams

Data Stream Processing analyze and process large volumes of streaming data with sub-millisecond latencies





- A data stream is an existing, continuous, ordered (implicitly by entrance time or explicitly by timestamp) chain of items. It is unfeasible to control the order in which units arrive, nor it is feasible to locally capture stream in its entirety.
- It is enormous volumes of data, items arrive at a high rate.

Data Streams**Types of Data Streams :**

Data stream –

A data stream is a(possibly unchained) sequence of tuples. Each tuple comprised of a set of attributes, similar to a row in a database table.

Transactional data stream –

It is a log interconnection between entities

Credit card – purchases by consumers from producer

Telecommunications – phone calls by callers to the dialed parties

Web – accesses by clients of information at servers

Measurement data streams –

Sensor Networks – a physical natural phenomenon, road traffic

IP Network – traffic at router interfaces

Earth climate – temperature, humidity level at weather stations

Characteristics of Data Streams :

1.Large volumes of continuous data, possibly infinite.
2.Steady changing and requires a fast, real-time response.
3.Data stream captures nicely real time processing needs of today.
4.Random access is expensive and a single scan algorithm
5.Store only the summary of the data seen so far.
6.Maximum stream data are at a pretty low level or multidimensional in creation, needs multilevel and multidimensional treatment.

Applications of Data Streams :

1.Fraud perception
2.Real-time goods dealing
3.Consumer enterprise
4.Observing and describing on inside IT systems

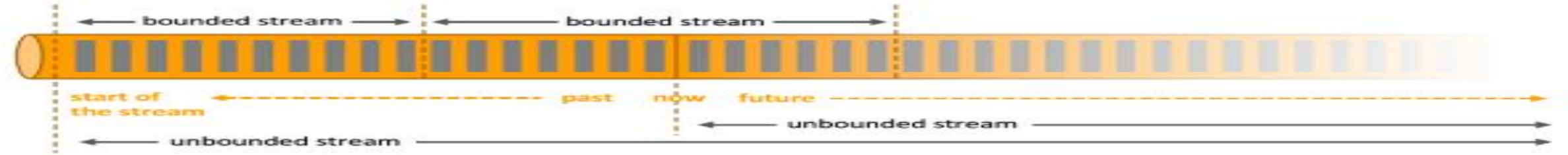
- Data streaming is the process of continuously collecting data as it's generated and moving it to a destination. This data is usually handled by stream processing software to analyze, store, and act on this information. Data streaming combined with stream processing produces real-time intelligence.
- Data streams can be created from various sources in any format and in any volume. The most powerful data streams aggregate multiple sources together

1. Bounded Stream

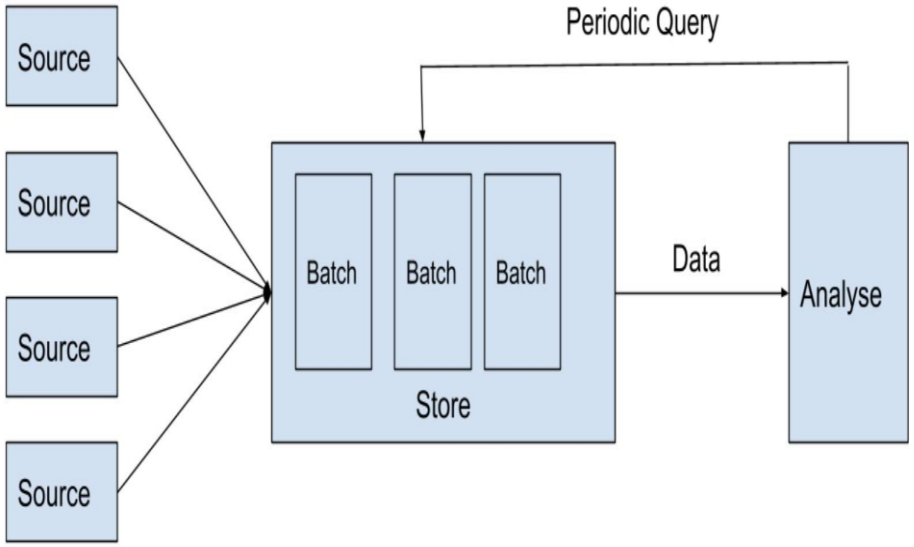
- The bounded stream will have a defined start and an end.
- While processing a bounded stream, the entire data-set can be ingested before starting any computation.
- The processing of bounded stream is referred to as **Batch processing**.

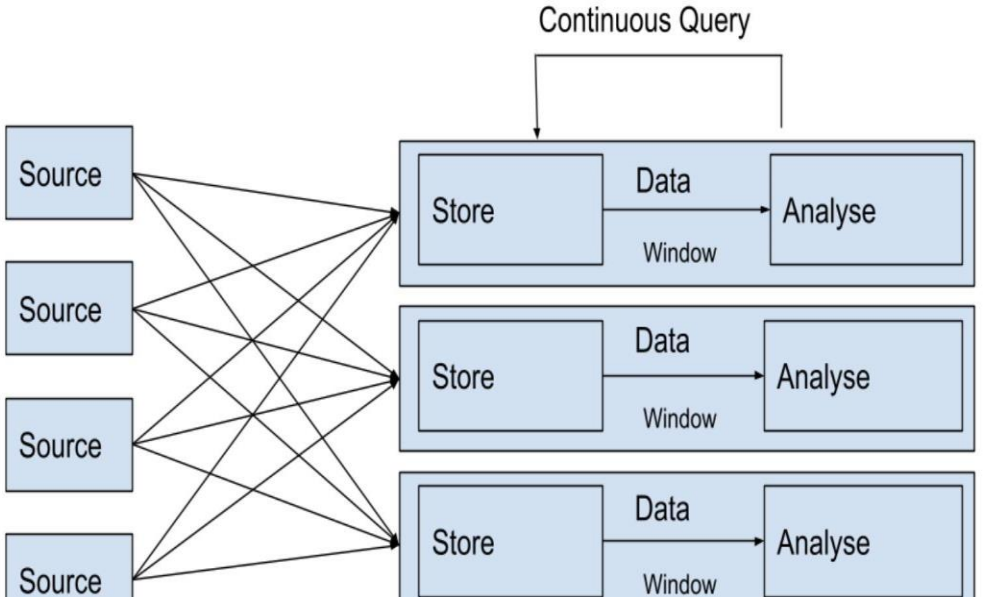
2. Unbounded Stream

- The unbounded stream will have a start but no end. Hence, the data needs to be continuously processed when it is generated. Data is processed based on the event time (Event time is the time that each individual event occurred on its producing device). The paradigm of processing unbounded stream is referred to as **Stream processing**.



Batch Processing	Stream Processing
Batch processing relates to the collection of large-scale non-continuous data and grouping them in batches. data at rest relies on batch processing	Stream processing is the collection and analysis of continuous streams of data immediately (a few seconds or milliseconds) after being generated. streaming data relies on stream processing
Batch processing processes the data after it has been generated and stored in a data warehouse	Stream processing processes the data right after it's collected from sources
Batch processing is more advantageous when data size is both known and finite	The rate of data output corresponds approximately to the rate of input
Batch processing is implemented in scenarios where real-time intelligence is not necessary or the data cannot be converted into a data stream for immediate analysis	Stream processing continuously ingests and analyzes data

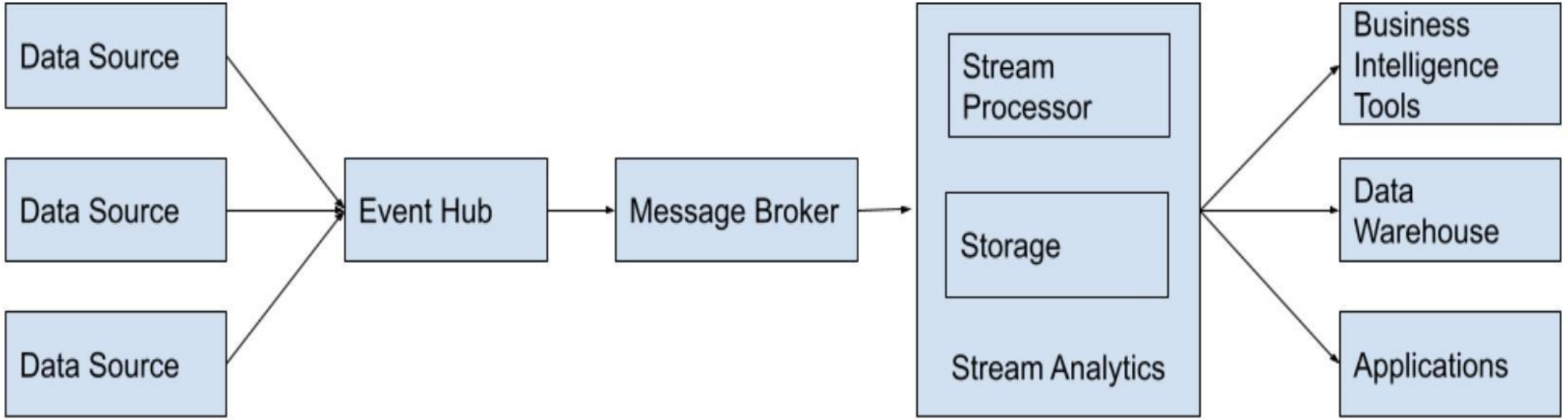




Stream Data Architecture

Streaming Data Architecture consists of software components, built and connected together to ingest and process streaming data from various sources.

- A standard data streaming architecture has three components:
- An event hub
- A message broker
- An analytics engine



Stream Data Architecture

- The **event hub** is the storage that collects different events from multiple data sources. It's a data ingestion service that makes sure data is collected sequentially.
- **Message brokers**, as the name suggests, collect data from sources, validate and translate it into a standard message format, and continuously send it to target destinations such as data warehouses or streaming analytics engines.
- Therefore, message brokers act as intermediaries and simplify communication between applications
- As streaming data is continuously created and analyzed, **stream analytics** has two stages that are conducted simultaneously: storage and processing.
- To support the ordering and consistency of the data, the storage stage must be able to support reads and writes of huge streams of data that are quick and repeatable
- Then, the processing stage consumes data from the storage stage, processes it, and finally communicates to the storage layer to delete records that are no longer required.
- After these steps, **data can be visualized through business intelligence tools or received by enterprise applications for consumption**.

Steps in query processing in Data Streams

Formulation of continuous queries

- The formulation of queries is mostly done **using** Continuous Query Language (CQL) and StreamSQL
- In StreamSQL, a query with a sliding window for the last 10 elements looks like follows:
- **SELECT AVG (price) FROM example stream [SIZE 10 ADVANCE 1 TUPLES] WHERE value > 100.0**

Translation of declarative query

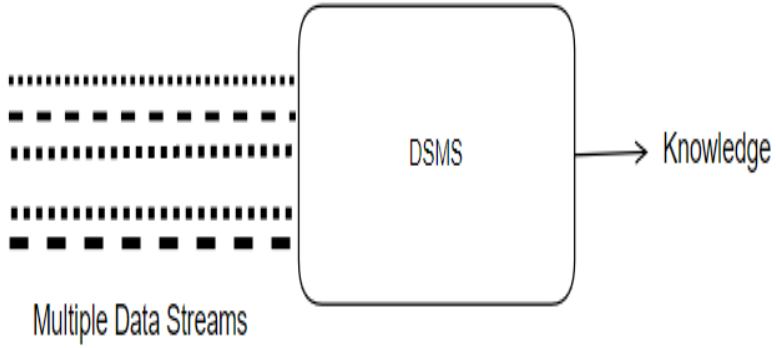
- The declarative query is translated into a logical query plan.
- A query plan is a directed graph where the nodes are operators and the edges describe the processing flow.
- Each operator in the query plan encapsulates the semantic of a specific operation, such as filtering or aggregation.

Optimization of queries :

- The logical query plan can be optimized, which strongly dependson the streaming model.
- E.g. If there are relational data streams and the logical query plan is based on relational operators from the Relational algebra, a query optimizer can use the algebraic equivalences to optimize the plan.

Compare Database Management System (DBMS) with Data Stream Management System (DSMS)

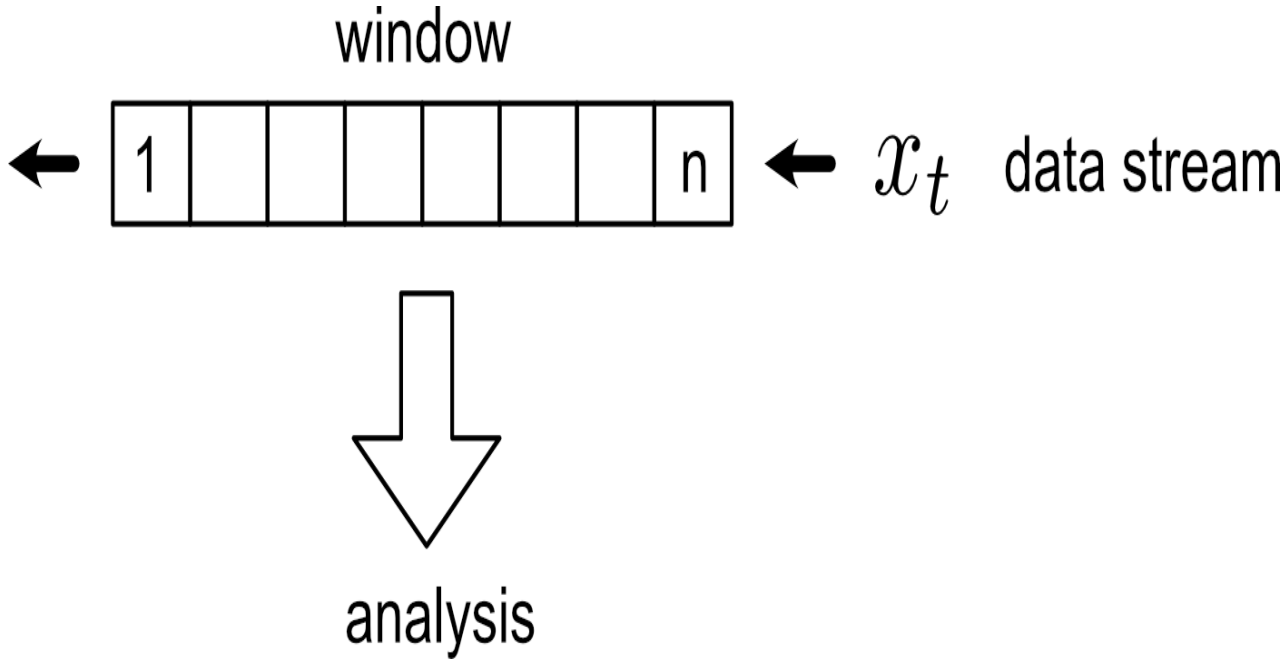
S. No.	Basis	DBMS	DSMS
1.	Data	Persistent relations	Streams, time windows
2.	Data access	Random	Sequential, one-pass
3.	Updates	Arbitrary	Append-only
4.	Update rates	Relatively low	High, bursty
5.	Processing model	Query driven (pull-based)	Data driven (push-based)
6.	Queries	One-time	Continuous
7.	Query plans	Fixed	Adaptive
8.	Query optimization	One query	Multi-query
9.	Query answer	Exact	Exact or approximate
10.	Latency	Relatively high	Low



Sampling data in a stream

- Sampling is a general technique for tackling massive amounts of data.
- Stream sampling is **the process of collecting a representative sample of the elements of a data stream**.
- The sample is usually much smaller than the entire stream, but can be designed to retain many important characteristics of the stream
- **Random Sampling : Sample m items randomly from the stream.**
- **Reservoir sampling**
- Reservoir sampling is a family of randomized algorithms for randomly choosing k samples from a list of n items, where n is either a very large or unknown number.
- A **simple solution** is to create an array reservoir[] of maximum size k. One by one randomly select an item from stream[0..n-1]. If the selected item is not previously selected, then put it in reservoir[]. To check if an item is previously selected or not, we need to search the item in reservoir[].
- **Reservoir sampling** is a randomized algorithm that is used to select k out of n samples; n is usually very large or unknown. This algorithm takes O(n) to select k elements with uniform probability.
- **Bernoulli sampling with its algorithm.**
- Bernoulli sampling The basic idea is that we add a record with probability $p \in (0, 1)$ to the set of samples.

```
int main() {
  // Defining the parameters
  int k = 4;
  int n = 8;
  // The array to be sampled
  int input[] = {1, 7, 4, 8, 2, 6, 5, 9};
  int output[k];
  // Getting a random seed every time
  srand (time(NULL));
  int i;
  // Initializing the output array to the first k elements
  // of the input array.
  for(i = 0; i < k; i++){
    output[i] = input[i];}
  int j;
  // Iterating over k to n-1
  for(j = i; j < n; j++){
    // Generating a random number from 0 to j
    int index = rand() % (j + 1);
    // Replacing an element in the  output with an element
    // in the input if the randomly generated number is less
    // than k.
    if(index < k){
      output[index] = input[j];
```



Counting Distinct Elements in a Stream

Flajolet-Martin Algorithm to Count the distinct elements in a stream.

- ✓ To estimate the number of unique elements appearing in a stream.
- ✓ Estimates unique elements in single pass
- ✓ If the stream contains n elements with m of them unique, this algorithm runs in O (n) time and needs O (log (m)) memory.
- ✓ Hash Function is utilized to narrow down the range of stream numbers
- ✓ Hash values (integers) are interpreted as binary numbers.
- ✓ 2 raised to the power that is the longest sequence of 0's seen in the hash value of any stream element is an estimate of the number of different elements.

- Stream: 4, 2, 5 ,9, 1, 6, 3, 7

- Hash function, h(x) = (ax + b) mod 32

- a) h(x) = 3x + 1 mod 32
  b) h(x) = x + 6 mod 32

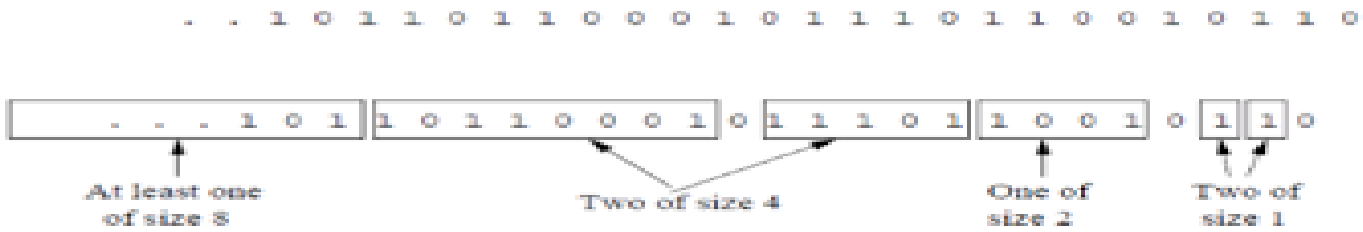
  a) h(x) = 3x + 7 mod 32
  h(4) = 3(4) + 7 mod 32 = 19 mod 32 = 19 = (10011)
  h(2) = 3(2) + 7 mod 32 = 13 mod 32 = 13 = (01101)
  h(5) = 3(5) + 7 mod 32 = 22 mod 32 = 22 = (10110)
  h(9) = 3(9) + 7 mod 32 = 34 mod 32 = 2 = (00010)
  h(1) = 3(1) + 7 mod 32 = 10 mod 32 = 10 = (01010)
  h(6) = 3(6) + 7 mod 32 = 25 mod 32 = 25 = (11001)
  h(3) = 3(3) + 7 mod 32 = 16 mod 32 = 16 = (10000)
  h(7) = 3(7) + 7 mod 32 = 28 mod 32 = 28 = (11100)
  Trailing zero's {0, 0, 1, 1, 1, 0, 4, 2}
  R = max [Trailing Zero] = 4
  Output = $2^R = 2^4 = 16$

DGIM Algorithm-Counting Oneness in window

- DGIM is **an efficient algorithm in processing large streams**. When it's infeasible to store the flowing binary stream, DGIM can estimate the number of 1-bits in the window.
- The windows are divided into buckets consisting of 1's and 0's.
- A bucket is a record consisting of timestamp
- Each bitstream has timestamp for the position it arrives

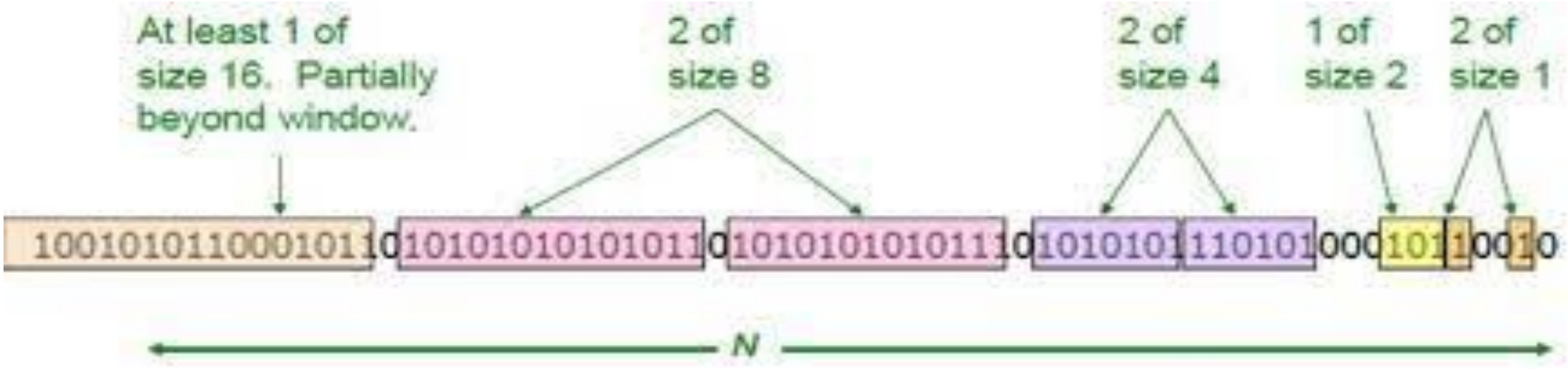
There are six rules that must be followed when representing a stream by buckets:

- The right end of a bucket is always a position with a 1.
- Every position with a 1 is in some bucket.
- No position is in more than one bucket.
- There are one or two buckets of any given size, up to some maximum size.
- All sizes must be a power of 2.
- Buckets cannot decrease in size as we move to the left (back in time).



RULES FOR FORMING THE BUCKETS:

1. The right side of the bucket should always start with 1. (if it starts with a 0,it is to be neglected) E.g. · 1001011 → a bucket of size 4 ,having four 1's and starting with 1 on it's right end.
2. Every bucket should have at least one 1, else no bucket can be formed.
3. All buckets should be in powers of 2.
4. The buckets cannot decrease in size as we move to the left. (move in increasing order towards left)

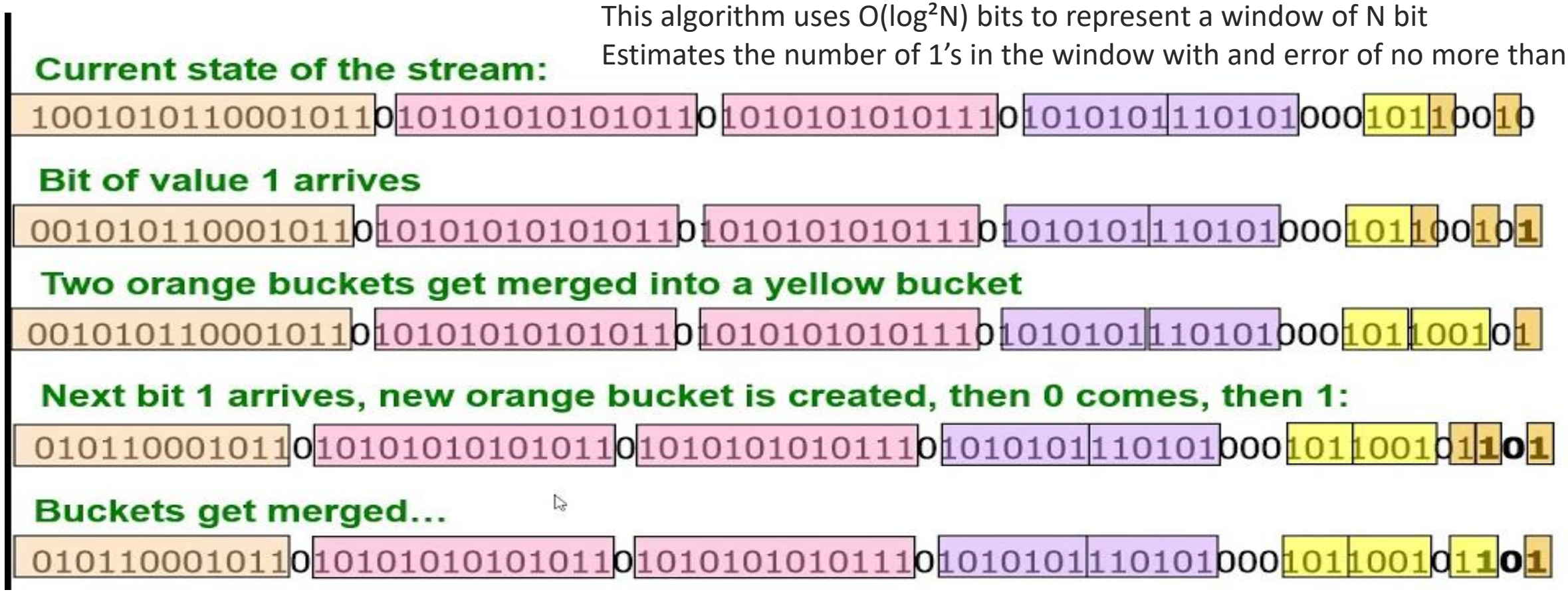


Updating Buckets

Buckets don’t overlap in timestamps

- If the current incoming bit is zero then no changes required in the buckets
- If the current bit is 1, set timestamp=current time
- If there are three buckets of size 1, then combine the oldest two into bucket of size two
- If there are three buckets of size 2, then combine oldest two into a bucket of size 4 and so on…
- To estimate the number of 1s in the window just add the size of buckets but not the last one
- Add half the size of last bucket

This algorithm uses $O(\log^2 N)$ bits to represent a window of N bit
Estimates the number of 1’s in the window with and error of no more than 50%.

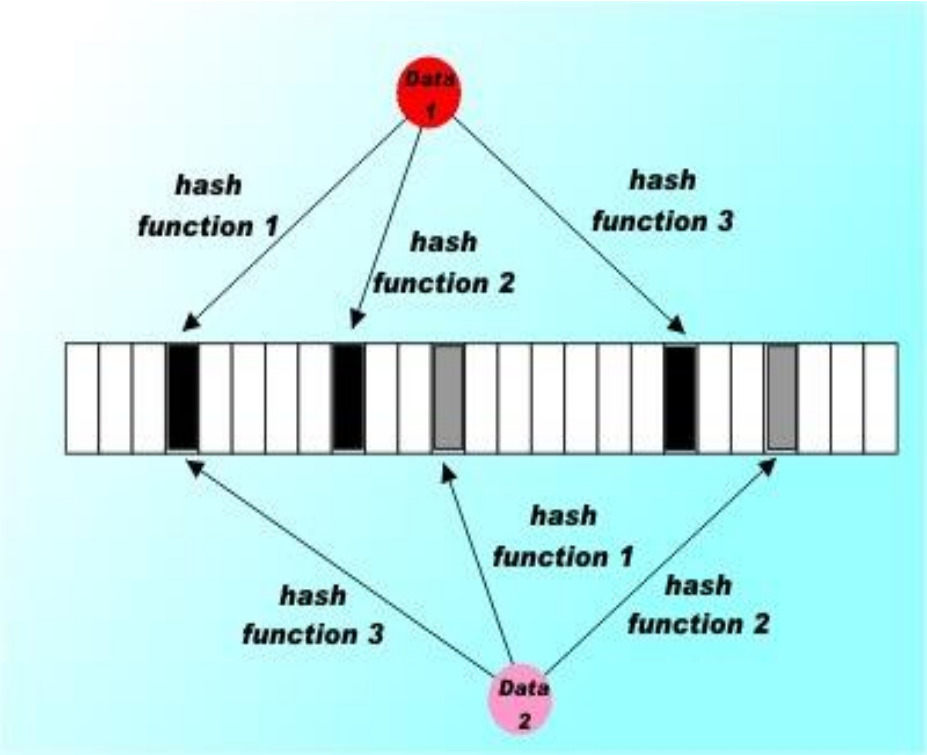


Data Filtering in Streams

- Data filtering is the process of choosing a smaller part of data set and using that subset for viewing or analysis.
- Filtering may be used to:
- Look at results for a particular period of time.
- Calculate results for particular groups of interest.
- Exclude erroneous or "bad" observations from an analysis.
- Train and validate statistical models.

Filtering Data Streams with Bloom Filter

- Bloom filters are for set membership which determines whether an element is present in a set or not.
- When a Bloom filter says an element is not present it is for sure not present. It guarantees 100% that the given element is not available in the set, because either of the bit of index given by hash functions will be set to 0.



The Bloom filter is just an array of bits with all bits set to zero initially.

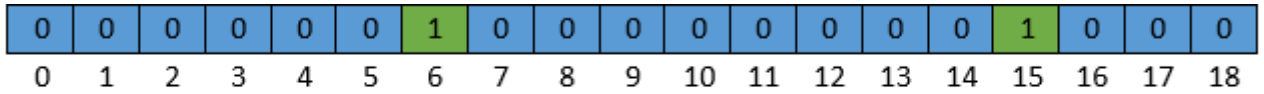
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

Bit Vector is implemented as a base data structure.

- Bloom filters support two actions
- First appending object and keeping track of an object
- Next verifying whether an object has been seen before.
- For appending objects to the Bloom filter
  - Compute hash values for the object to append
  - Hash value is the position of the bit to set
- Verifying whether the Bloom filter contains an object
  - Compute hash values for the object to append
  - Next verify whether the bits indexed by these hash values are set in the Bloom filter state.

Each hash function simply determines which bit to set or to verify.

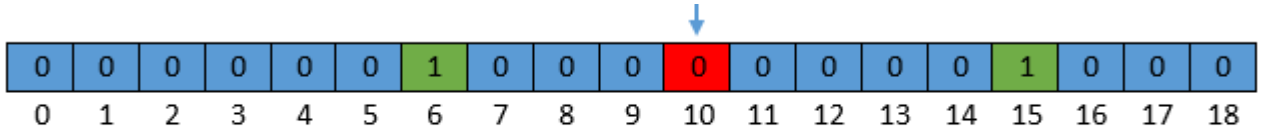
- E.g. For 1355 and 8337, respectively, the modulo operation is performed to get an index within the bounds of the bit array: 1355%19 = 6 and 8337%19 = 15.



To verify that bloom filter contain certain objects or not

1.Hash the input value
2.Mod the result by the length of the array
3.Check if the corresponding bit is 0 or 1

If the bit is 0, then that input definitely isn't a member of the set. But if the bit is 1, then that input might be a member of a set.



Limitations of the Bloom Filter

Though the Bloom filter is a space and time-efficient data structure, there are a few limitations of it:

•The naive implementation of the Bloom filter doesn't support the delete operation
•The *false-positives* rate can be reduced but can't be reduced to zero

Decaying Window

- With the streaming data, it is almost impossible to store sufficient statistics all over the examples, because of the memory constraints.
- With time, the properties of the datasets might change and the older data may not give the most important information for a statistical learning algorithm.
- The changes in the properties of data over time is called concept drift.

Case studies – real
time sentiment analysis, stock market predictions