Streaming Queries

CSE 4/562: Database Systems | Lecture 16

Sequential Data

Sequential Data


```
CREATE TABLE Transactions(transaction_date date,
                          product_id int,
                          price decimal);
```

Find the % change in monthly sales, each month

```
CREATE TABLE Transactions(transaction_date date,
                          product_id int,
                          price decimal);
```

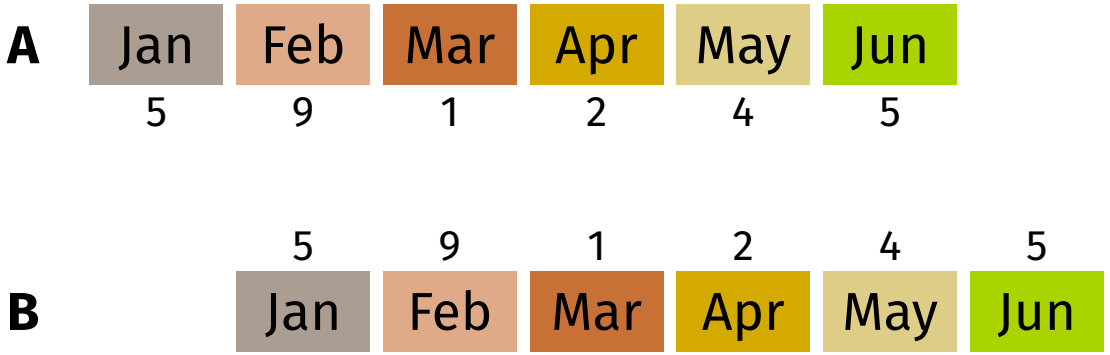
Find the % change in monthly sales, each month

```
SELECT A.Month, (A.Sales-B.Sales) / B.Sales
 FROM (SELECT … AS Month, SUM(…) AS Sales FROM …) A,
      (SELECT … AS Month, SUM(…) AS Sales FROM …) B
WHERE A.Month = B.Month + 1
```

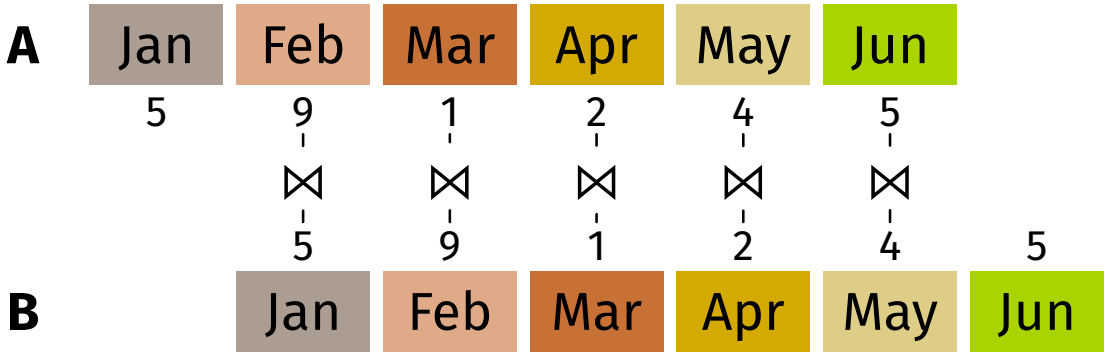
Find the % change in monthly sales, each month

	Jan	Feb	Mar	Apr	May	Jun
A	5	9	1	2	4	5

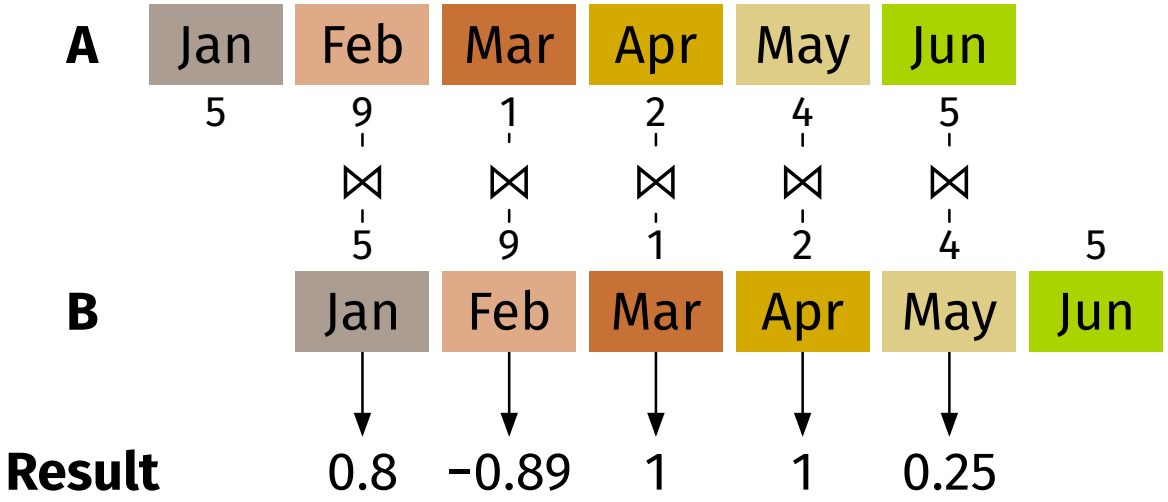
Find the % change in monthly sales, each month



Find the % change in monthly sales, each month



Find the % change in monthly sales, each month



```
CREATE TABLE Transactions(transaction_date date,
                          product_id int,
                          price decimal);
```

Find the daily top-5 products by sales in the last week

```
CREATE TABLE Transactions(transaction_date date,
                          product_id int,
                          price decimal);
```

Find the daily top-5 products by sales in the last week

```
SELECT product_id, SUM(…) AS Sales FROM … WHERE … = today - 1
ORDER BY Sales Desc LIMIT 5 UNION ALL
SELECT product_id, SUM(…) AS Sales FROM … WHERE … = today - 2
ORDER BY Sales Desc LIMIT 5 UNION ALL …
```

```
CREATE TABLE Transactions(transaction_date date,
                          product_id int,
                          price decimal);
```

Find the trailing n-day moving average of sales

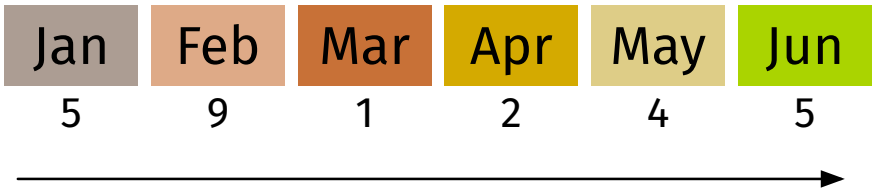
```
CREATE TABLE Transactions(transaction_date date,
                          product_id int,
                          price decimal);
```

Find the trailing n-day moving average of sales

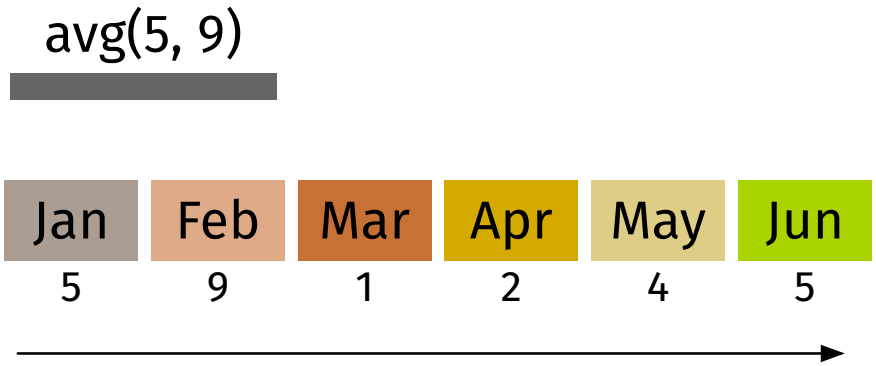
> ... almost impossible to express if n is a parameter (i.e., query size depends on N)

The WINDOW Operator

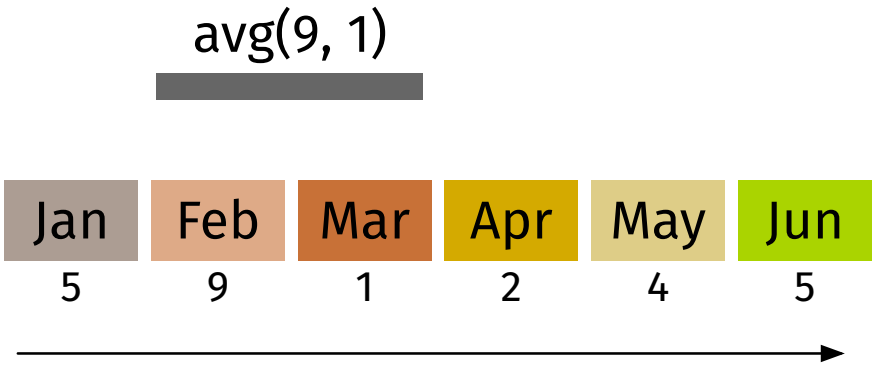
The WINDOW Operator



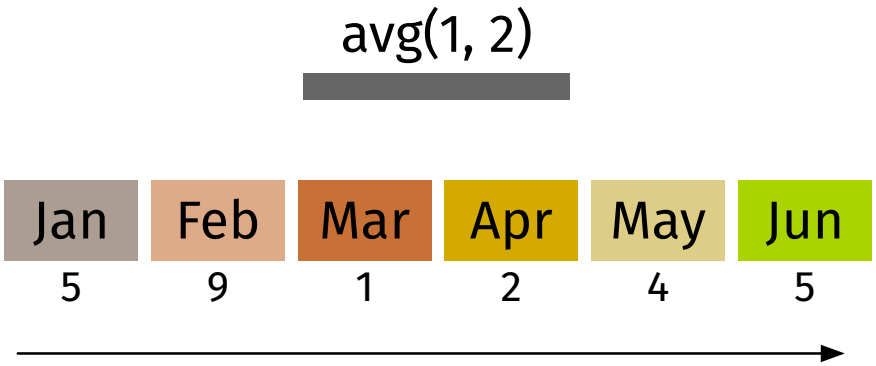
The WINDOW Operator



The WINDOW Operator



The WINDOW Operator



Windows

1. Partition the records into bins
2. Sort the records in each bin
3. Construct a window for each record consisting of records...
   - ... in the same bin
   - ... preceding the record within a given range
   - ... or following the record within a given range
4. Compute an aggregate for each window

Windows

1. Partition the records into bins
2. Sort the records in each bin
3. Construct a window for each record consisting of records...
   - ... in the same bin
   - ... preceding the record within a given range
   - ... or following the record within a given range
4. Compute an aggregate for each window

Ranges

- **Physical Size**: N records exactly.
- **Logical Size**: Records within a given range on the sort order.

WINDOW Queries

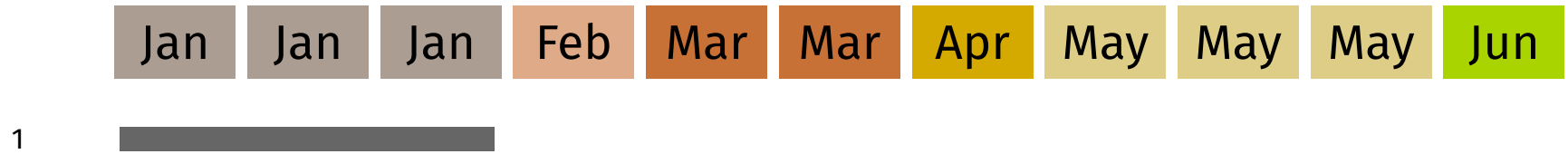

```
SELECT ...,  AGG(...) OVER W
...
WINDOW W AS (
  PARTITION BY attr
  ORDER BY attr
  RANGE BETWEEN range PRECEDING
        AND range FOLLOWING
)
```

WINDOW Queries


```
SELECT L.state, T.month,
   AVG(S.sales) OVER W as movavg
FROM   Sales S, Times T, Locations L
WHERE  S.timeid = T.timeid
  AND  S.locid = L.locid
WINDOW W AS (
   PARTITION BY L.state
   ORDER BY T.month
   RANGE BETWEEN INTERVAL ‘2’ MONTH PRECEDING
         AND INTERVAL ‘0’ MONTH FOLLOWING
)
```

WINDOW Queries

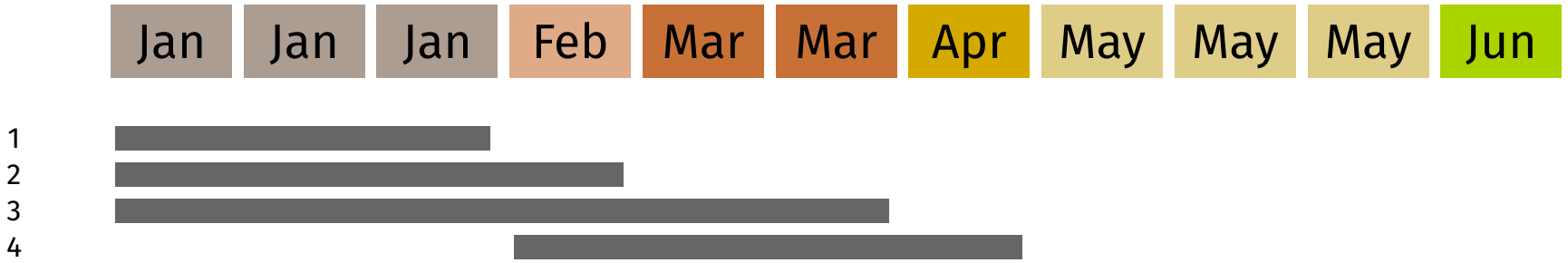
WINDOW Queries



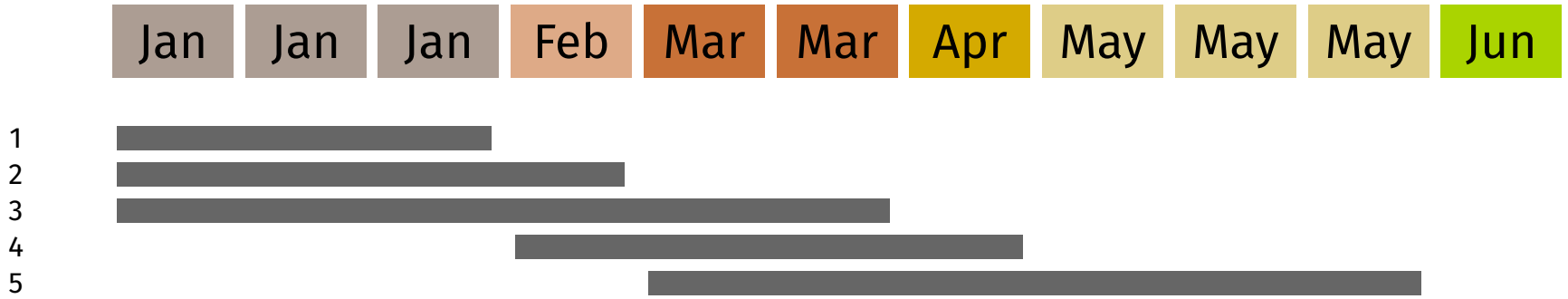
WINDOW Queries

WINDOW Queries

WINDOW Queries



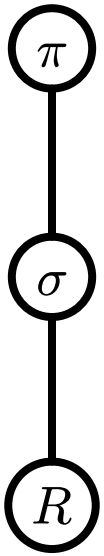
WINDOW Queries



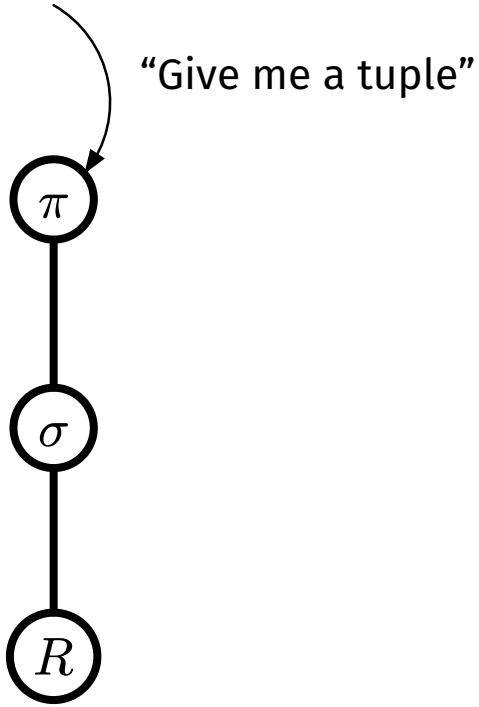
Streaming Queries

Challenge: Standing queries react to new data *as it arrives*

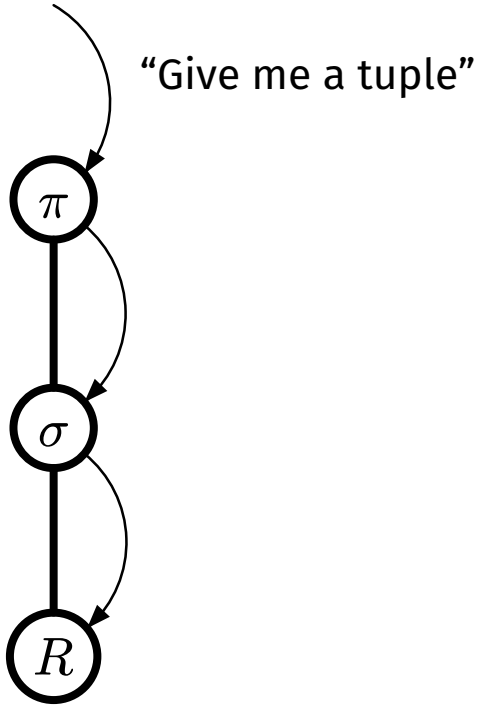
Push vs Pull - Pull



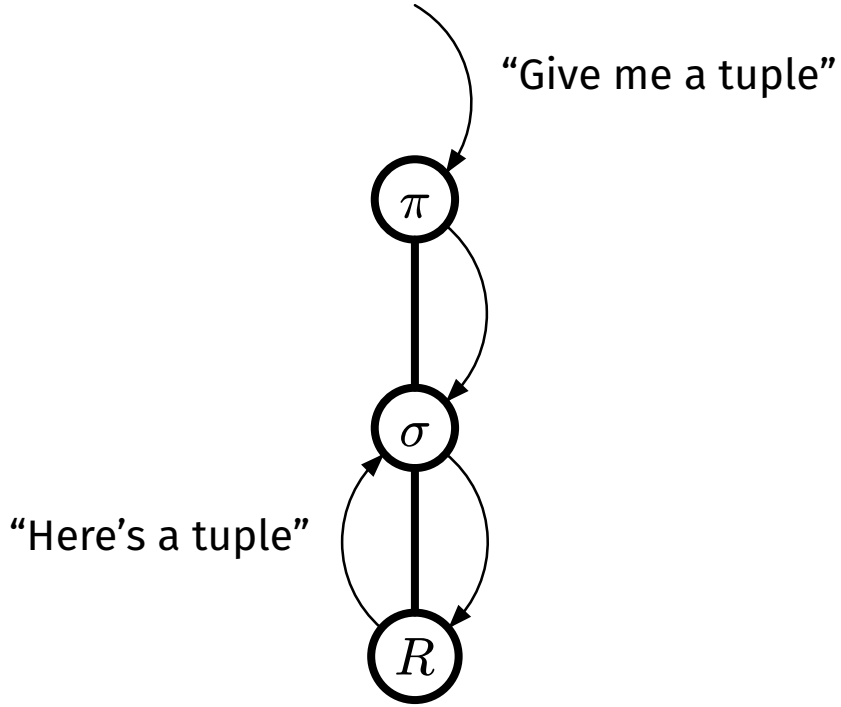
Push vs Pull - Pull



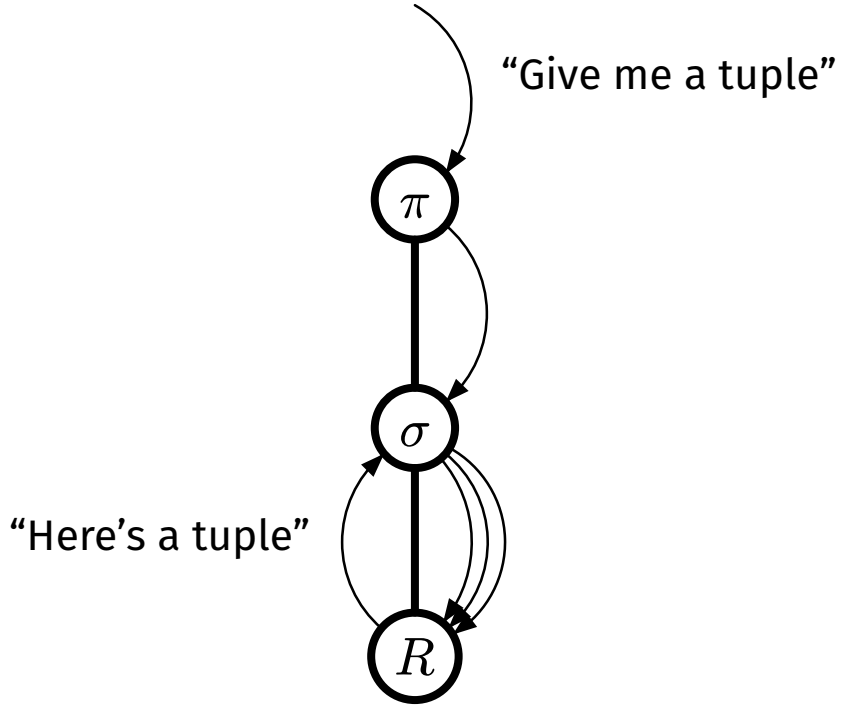
Push vs Pull - Pull



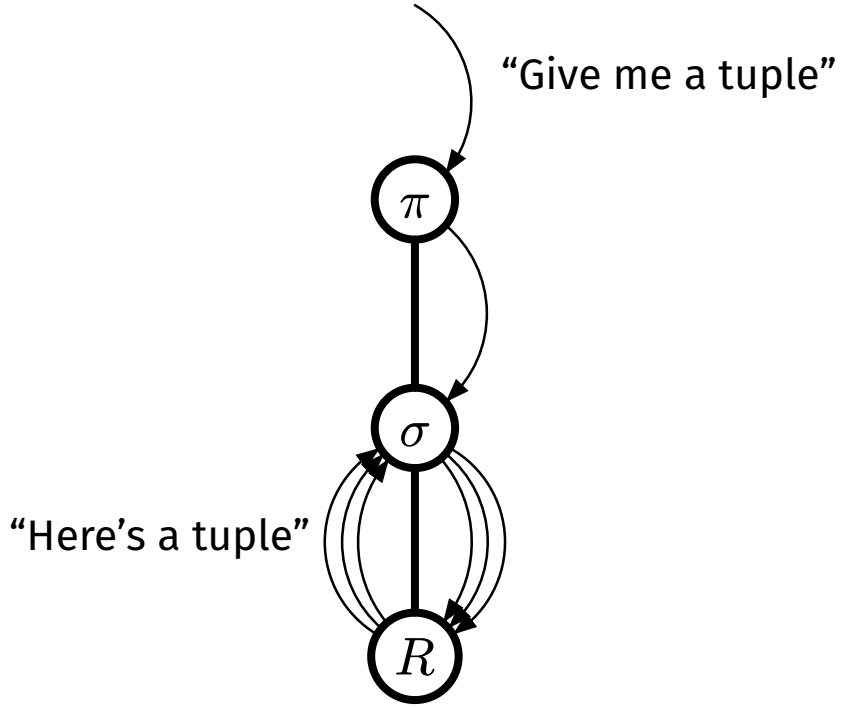
Push vs Pull - Pull



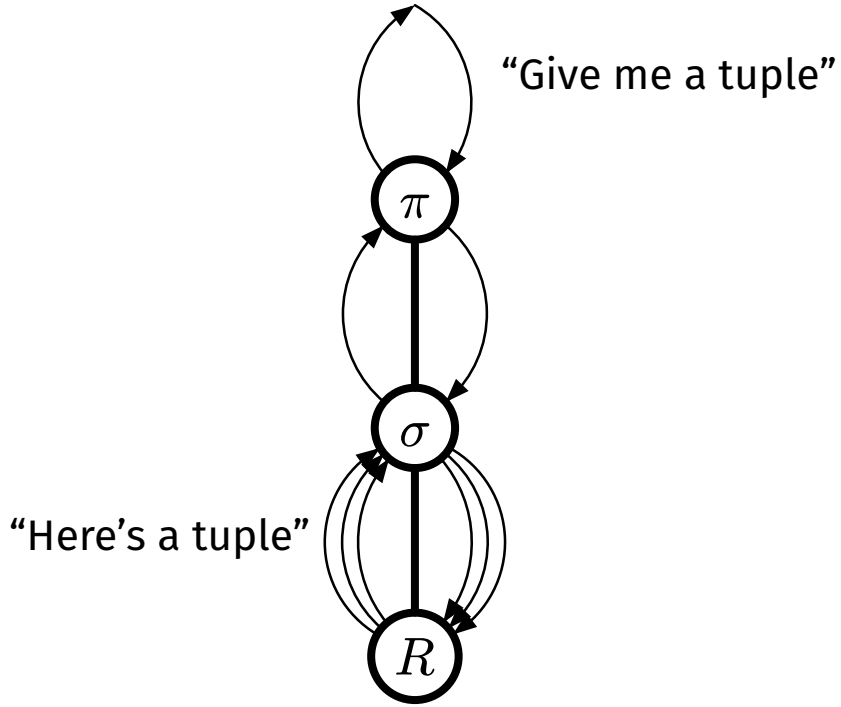
Push vs Pull - Pull



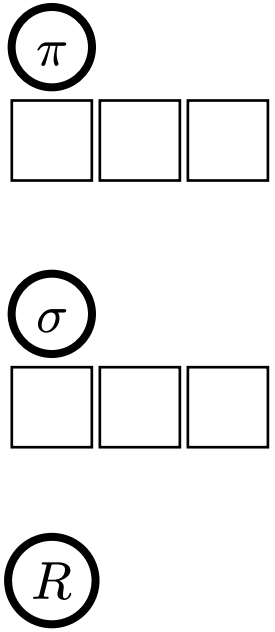
Push vs Pull - Pull



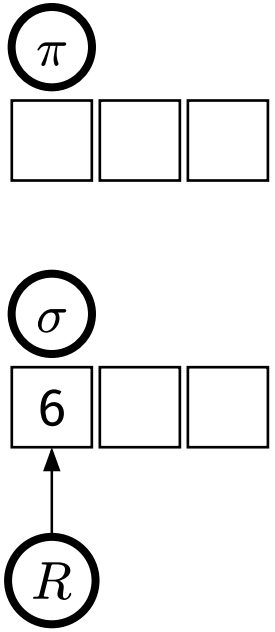
Push vs Pull - Pull



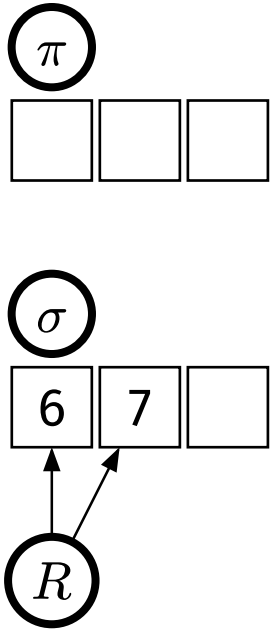
Push vs Pull - Push



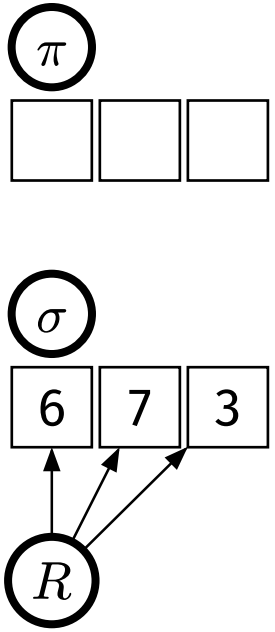
Push vs Pull - Push



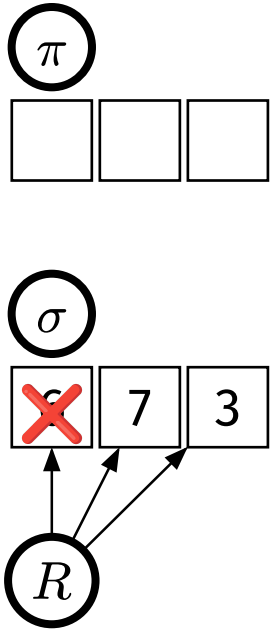
Push vs Pull - Push



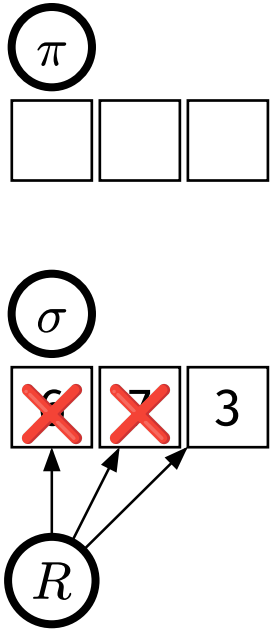
Push vs Pull - Push



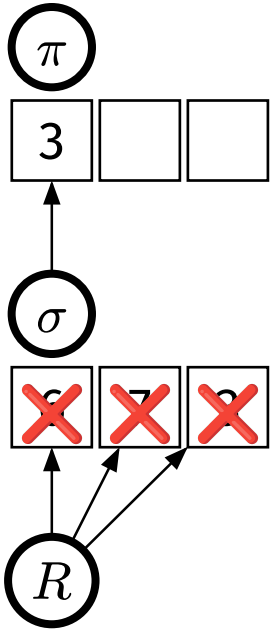
Push vs Pull - Push



Push vs Pull - Push



Push vs Pull - Push

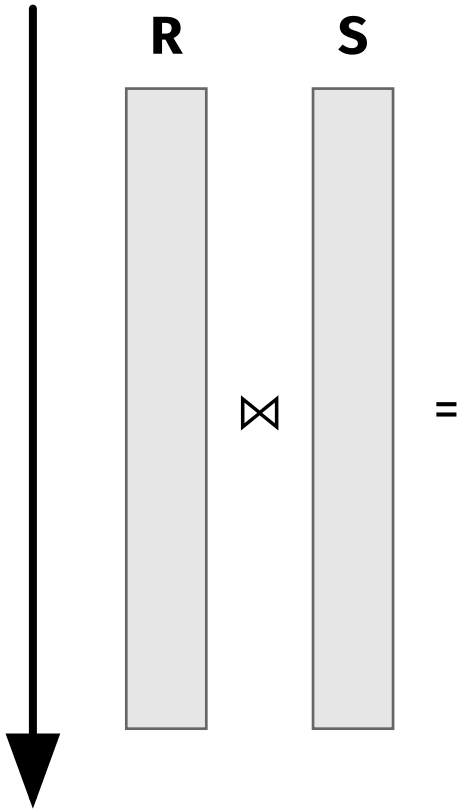


Push Operators

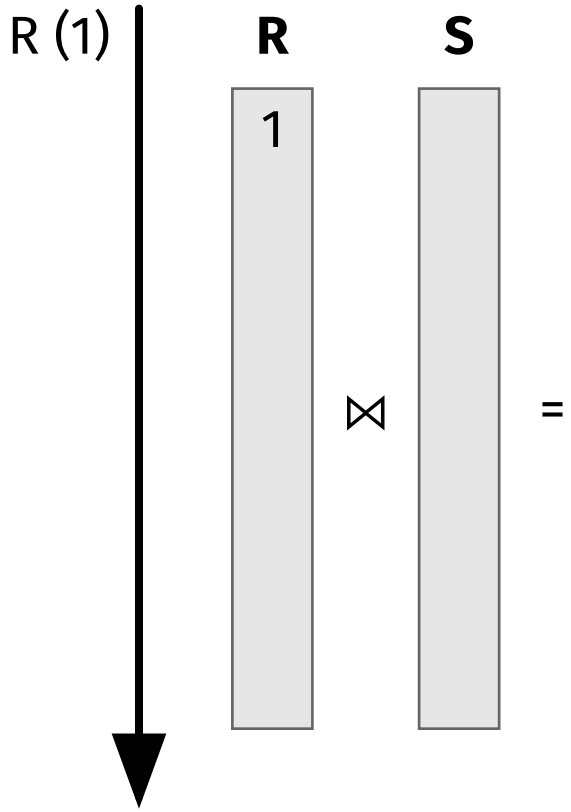
- Each operator acts independently
- Each operator buffers input
- Operators are scheduled

Streaming Joins

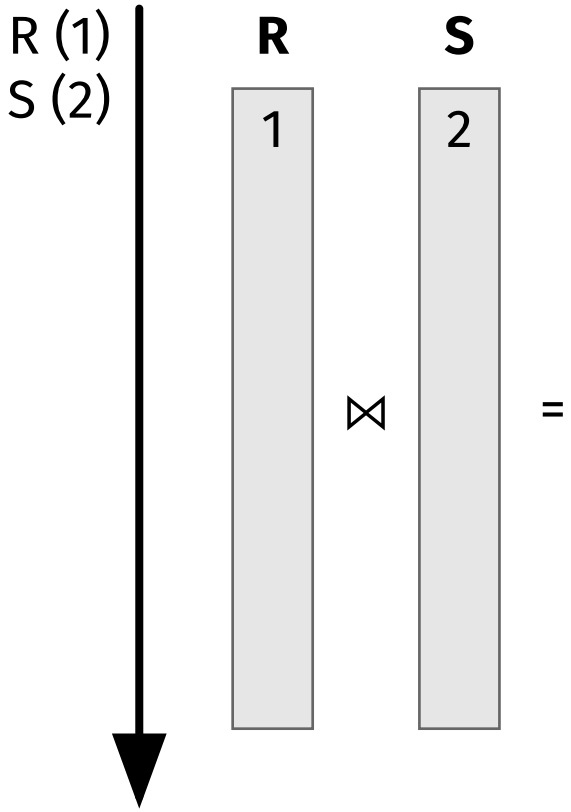
Streaming Join



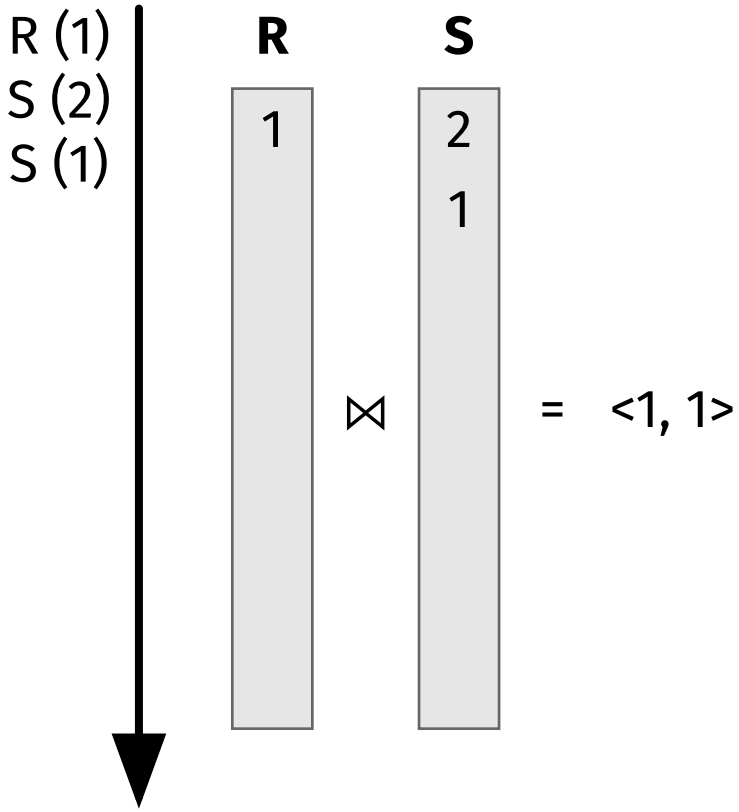
Streaming Join



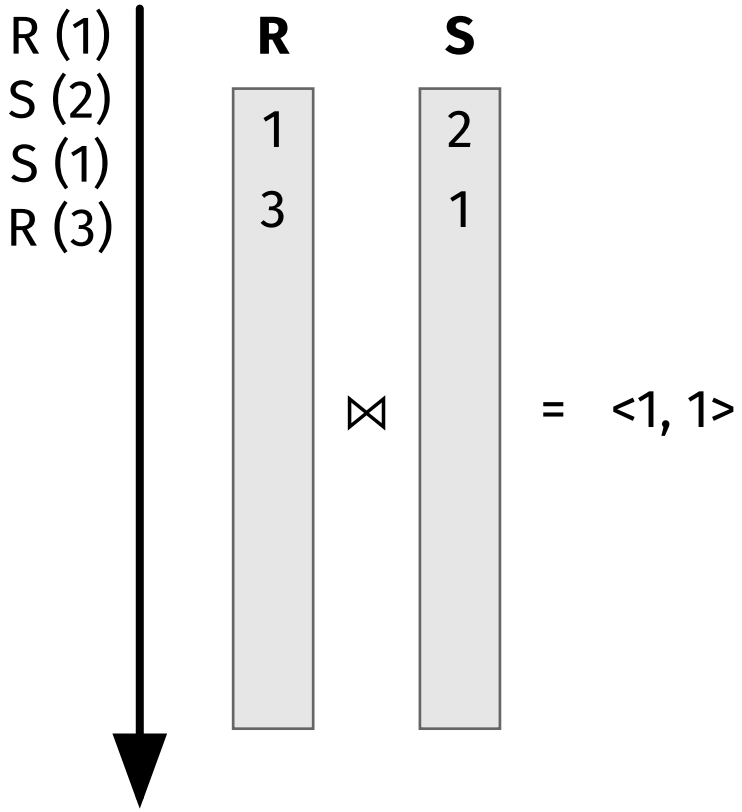
Streaming Join



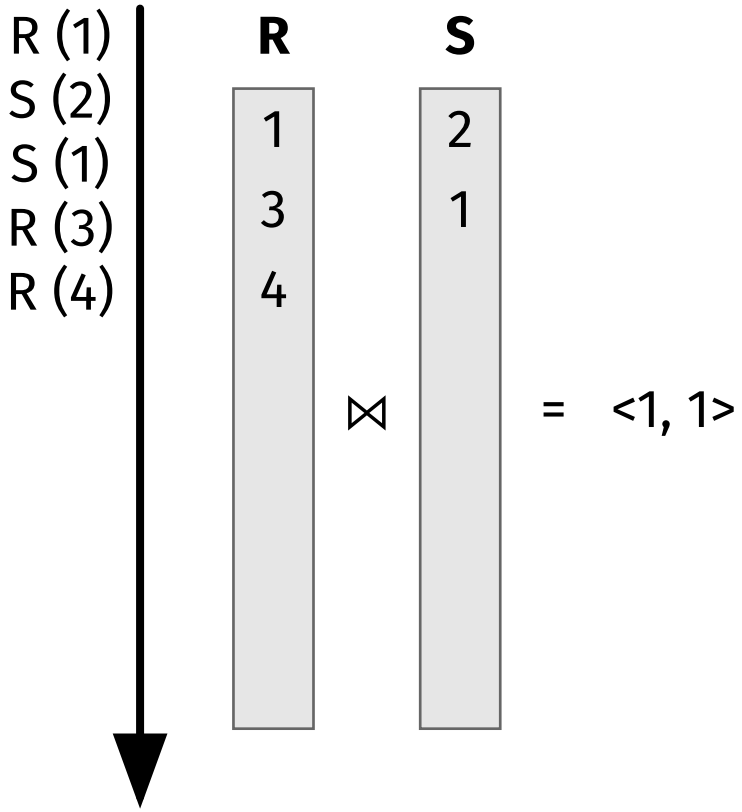
Streaming Join



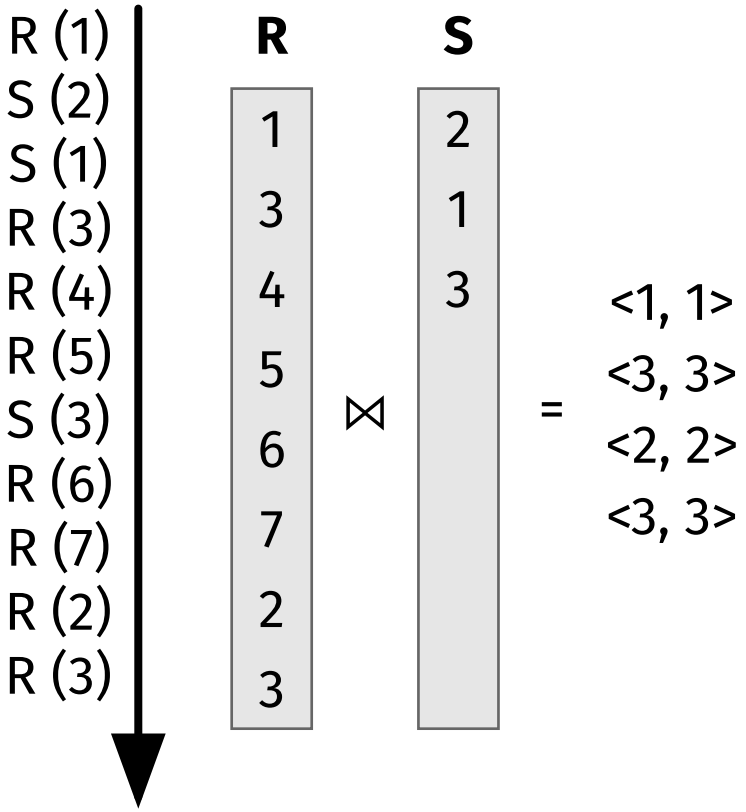
Streaming Join



Streaming Join



Streaming Join



- **Problem 1**: You eventually run out of memory.
- **Problem 2**: The cost of iterating over the buffer keeps growing.

Streaming Joins

- **Idea 1**: Mandate only `WINDOW` queries

Streaming Joins

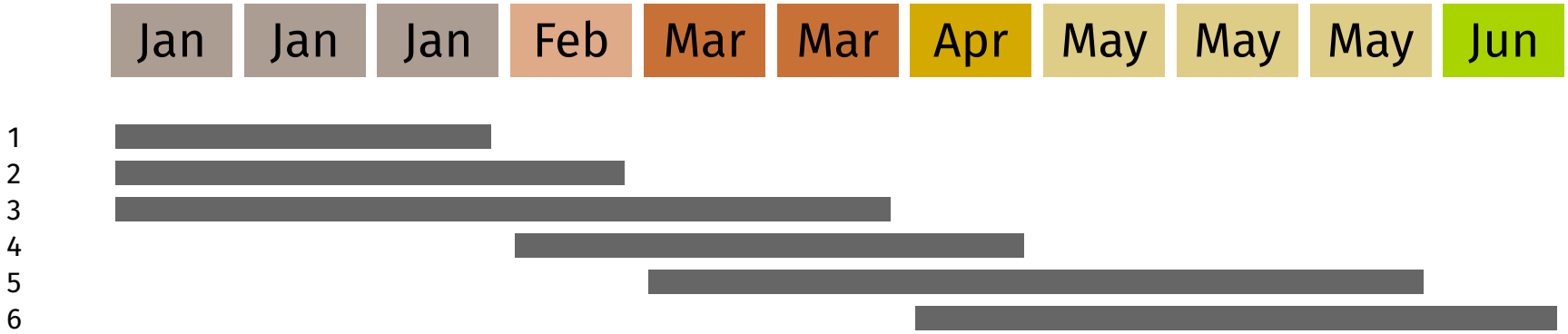
- **Idea 1**: Mandate only `WINDOW` queries
- **Idea 2**: Index the buffer

- **Idea 1**: Mandate only `WINDOW` queries
- **Idea 2**: Index the buffer

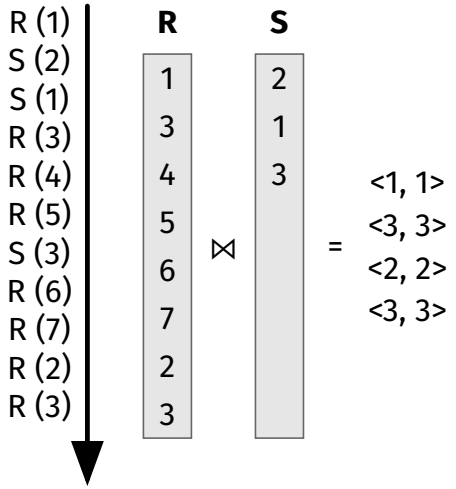
Challenge: How do we maintain the index as tuples fall out of the buffer?

Streaming Indexes

Streaming Indexes



Tuples always enter from one side and exit out the other



Lots of lookups for active tuples

How do we organize the data?

How do we organize the data?

Queue/Linked List

- Insert/remove in (nearly) temporal order

How do we organize the data?

Queue/Linked List

- Insert/remove in (nearly) temporal order

Hash/Tree

- Efficient lookup for (randomly ordered) join keys

Streaming Indexes

<1, **1**>

<2, **3**>

<3, **4**>

<4, **3**>

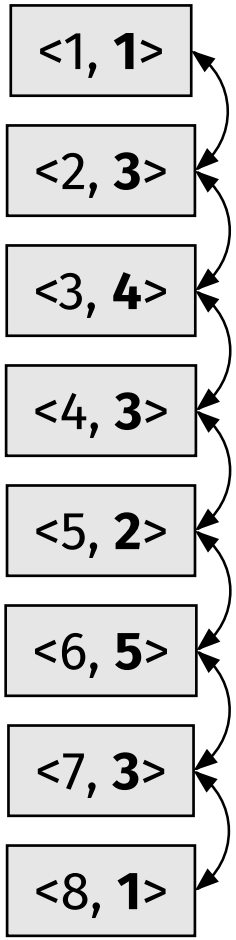
<5, **2**>

<6, **5**>

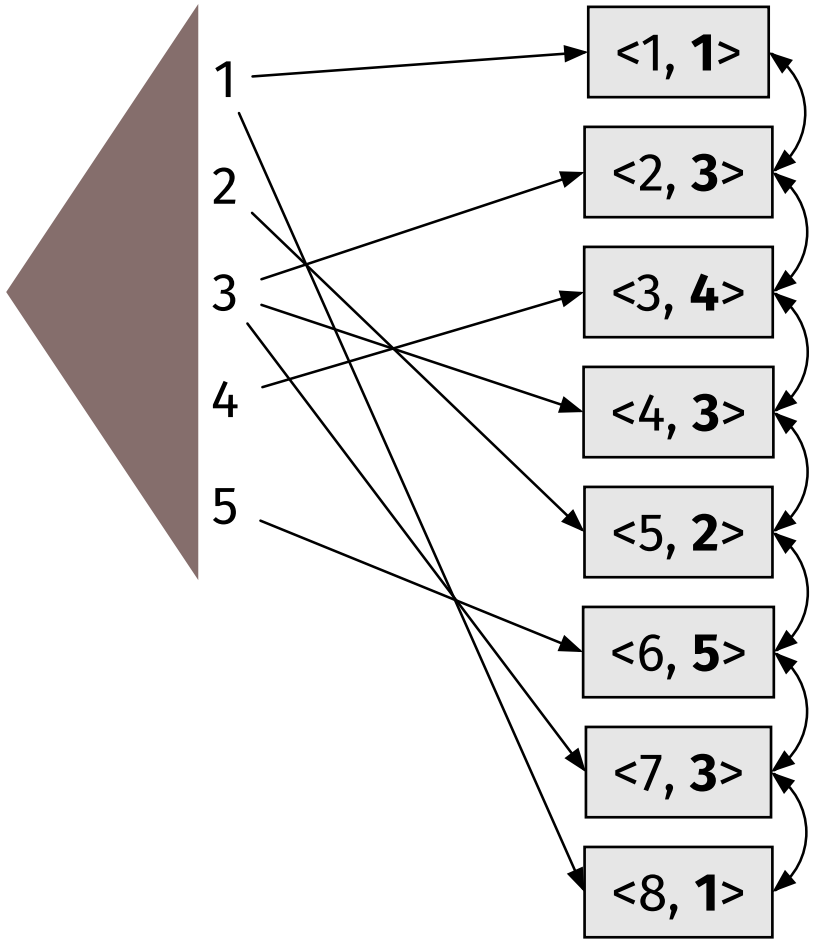
<7, **3**>

<8, **1**>

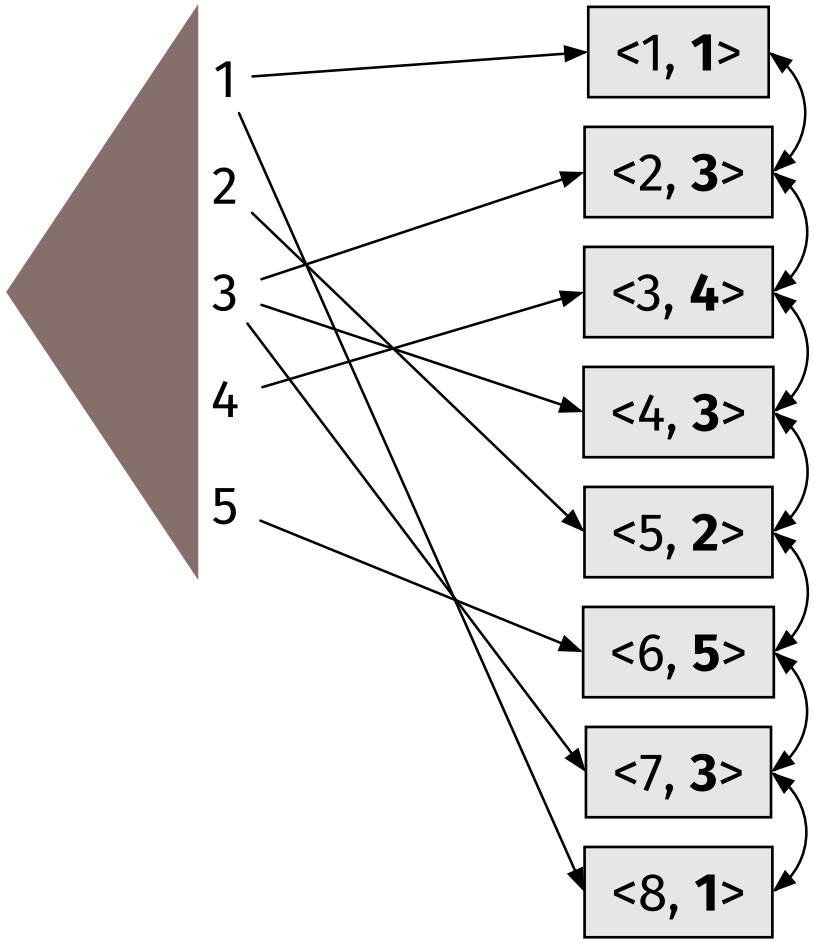
Streaming Indexes



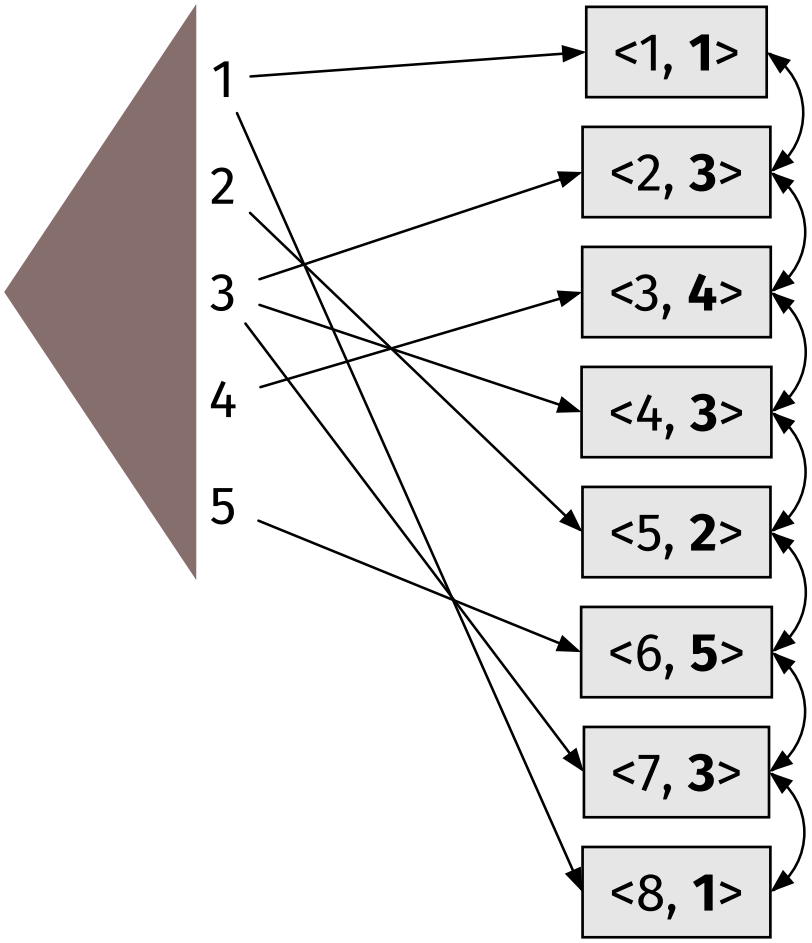
Streaming Indexes



Streaming Indexes



Streaming Indexes



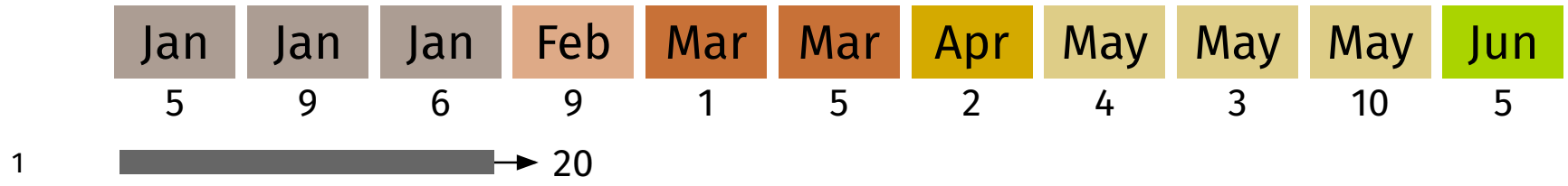
Streaming Indexes

$O(1 + \log(|W|))$ per insertion.

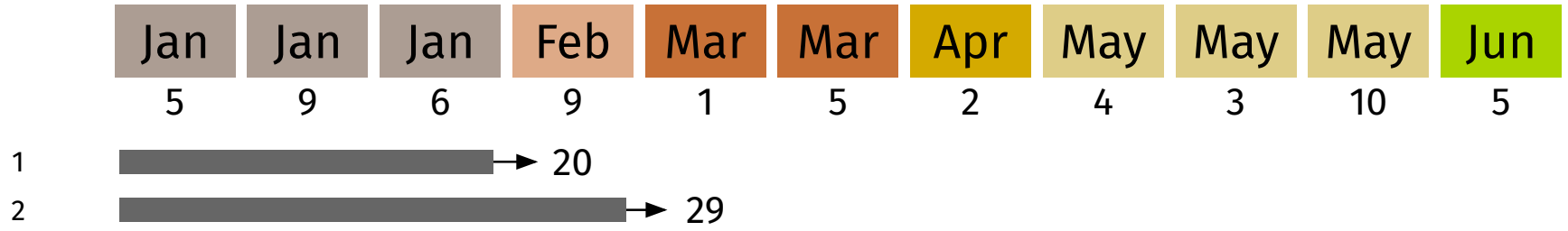
$O(1 + \log(|W|))$ per expiration.

WINDOW Indexes - SUM

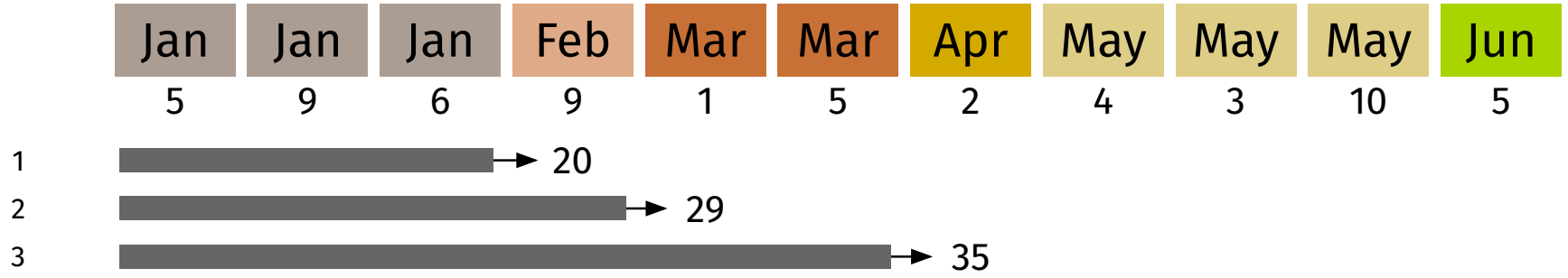
WINDOW Indexes - SUM



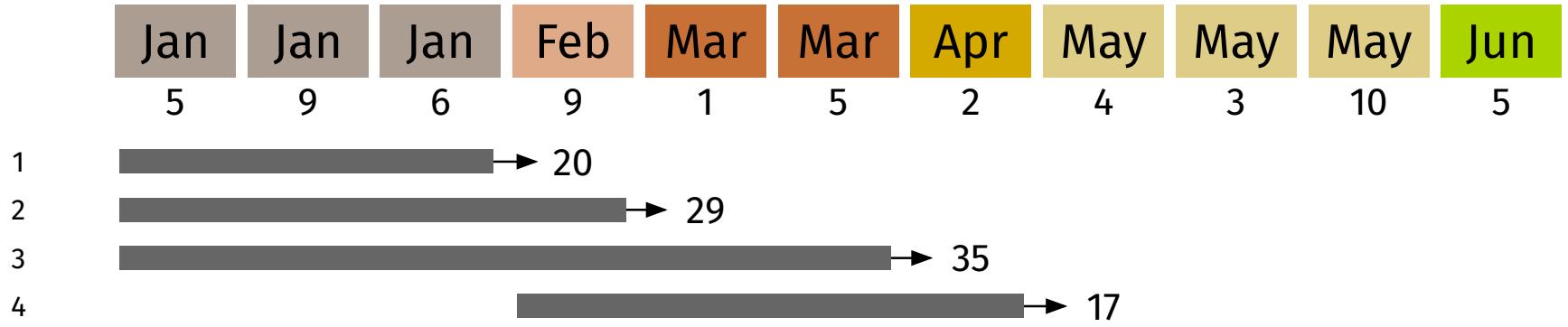
WINDOW Indexes - SUM



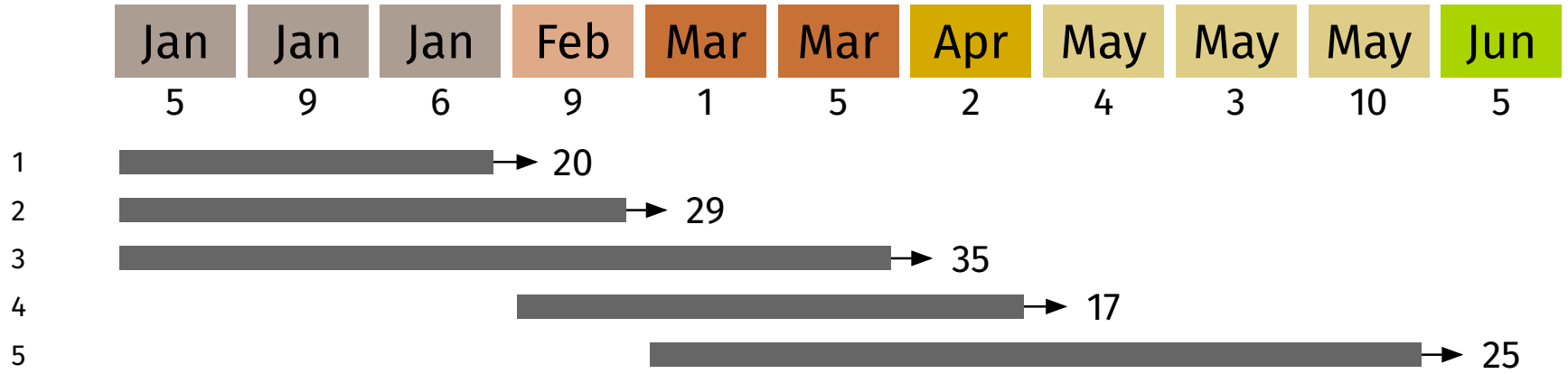
WINDOW Indexes - SUM



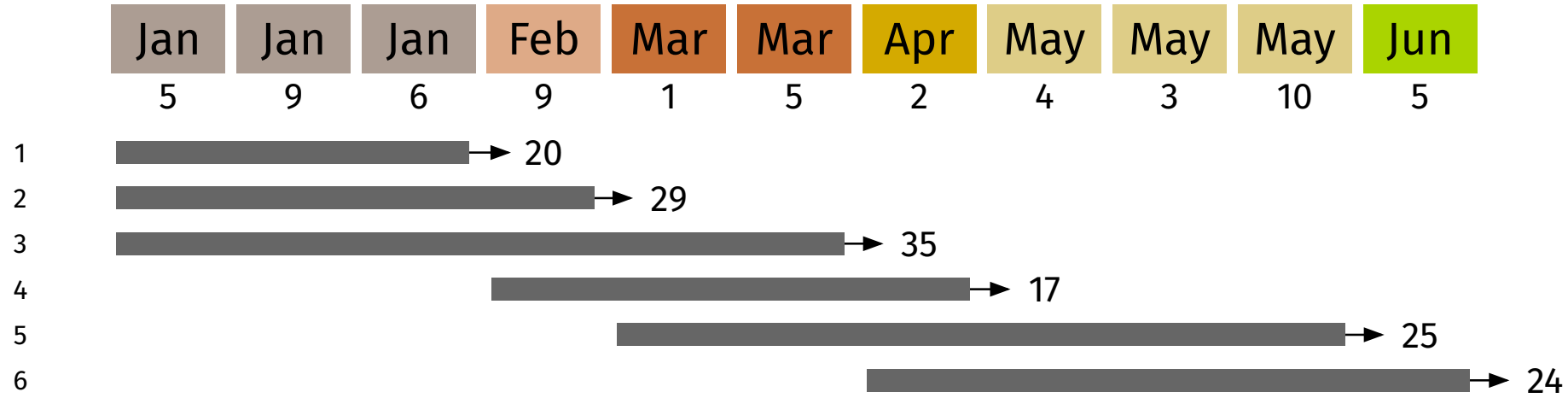
WINDOW Indexes - SUM



WINDOW Indexes - SUM



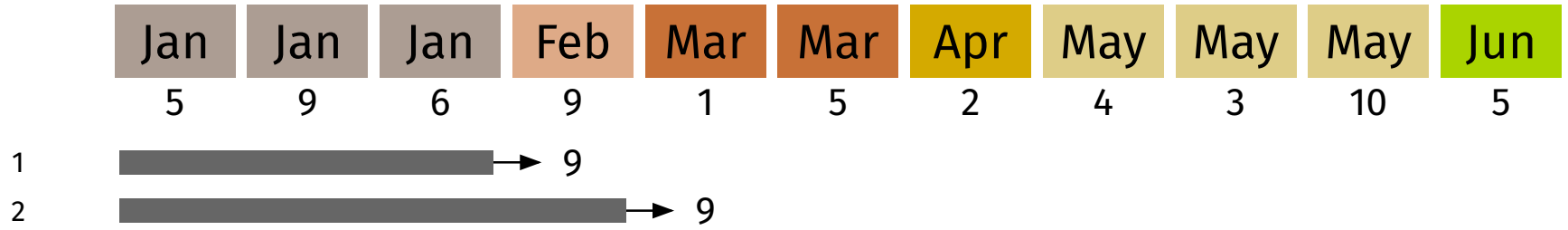
WINDOW Indexes - SUM



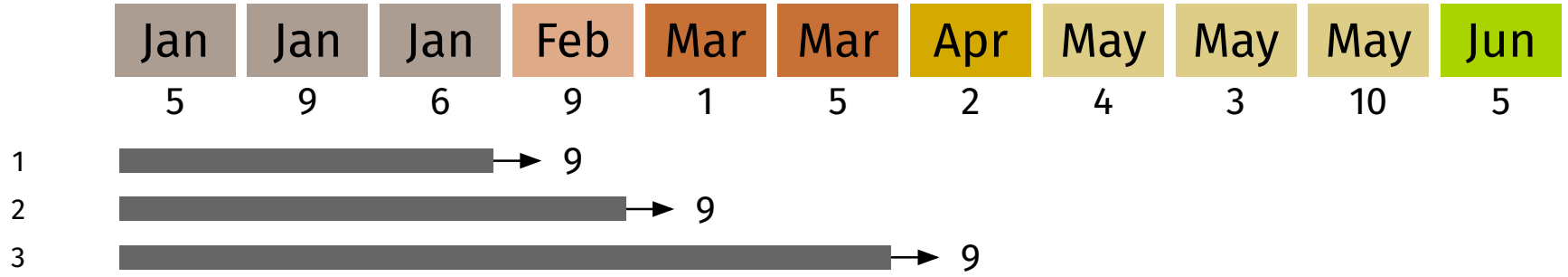
WINDOW Indexes - MAX

WINDOW Indexes - MAX

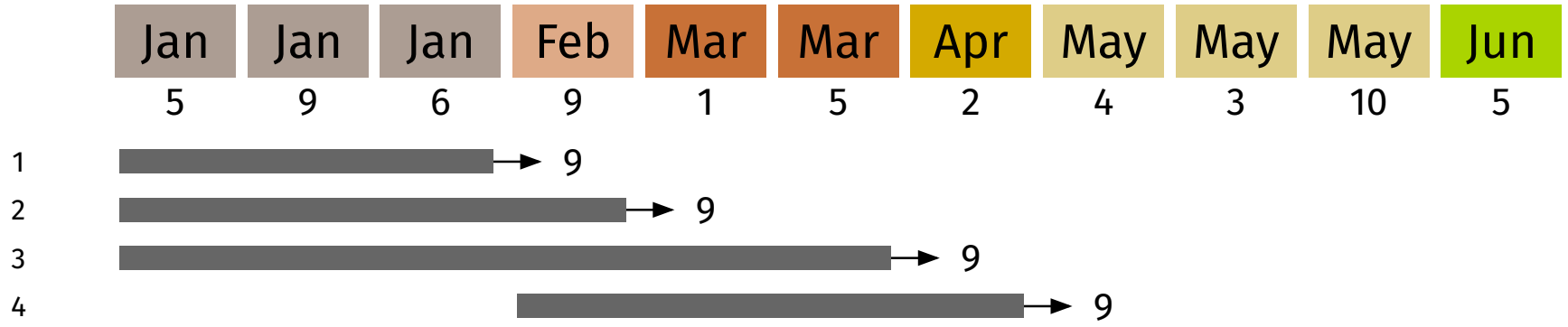
WINDOW Indexes - MAX



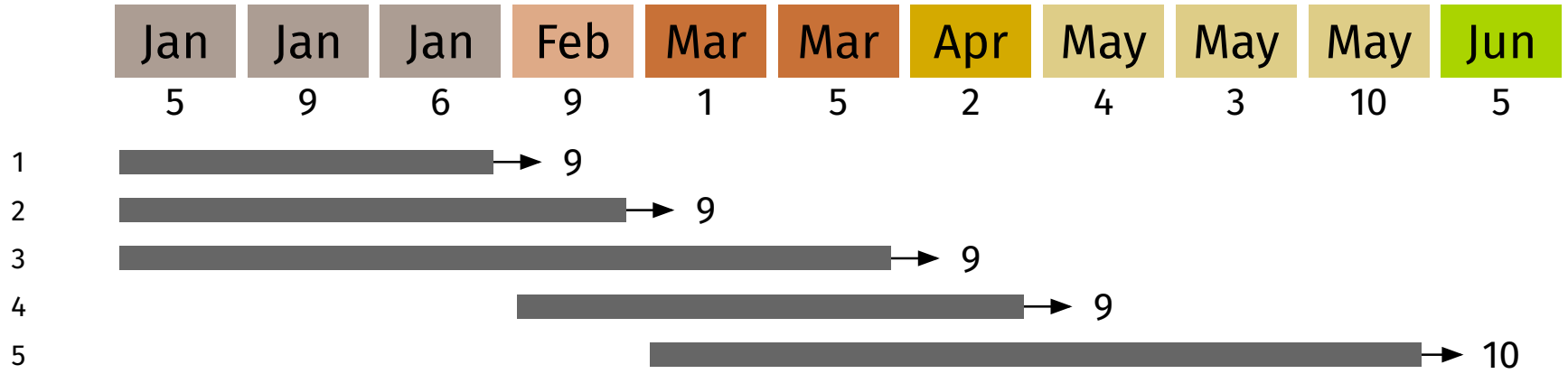
WINDOW Indexes - MAX



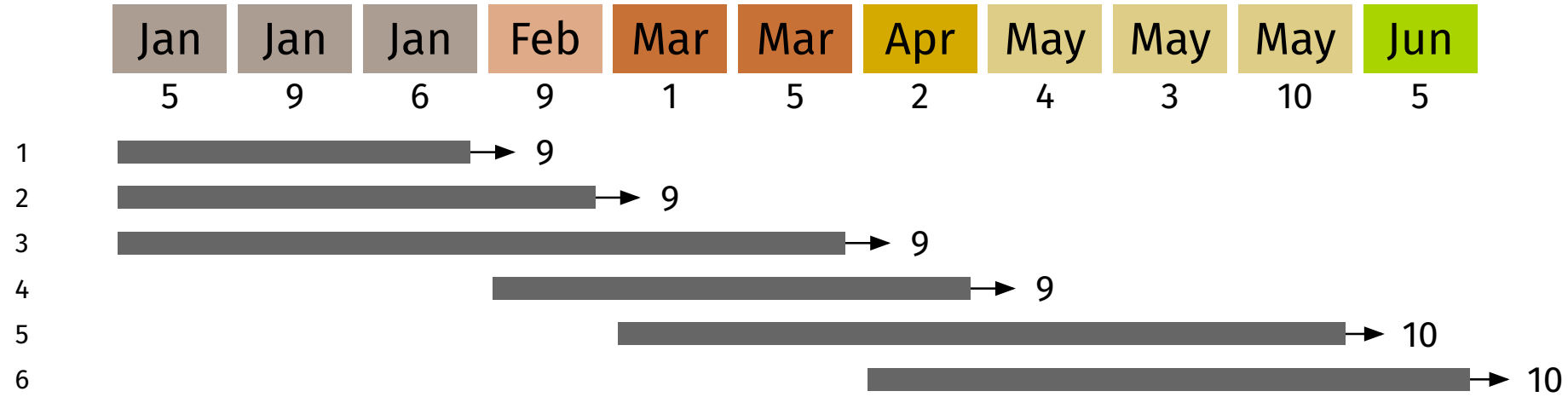
WINDOW Indexes - MAX



WINDOW Indexes - MAX



WINDOW Indexes - MAX



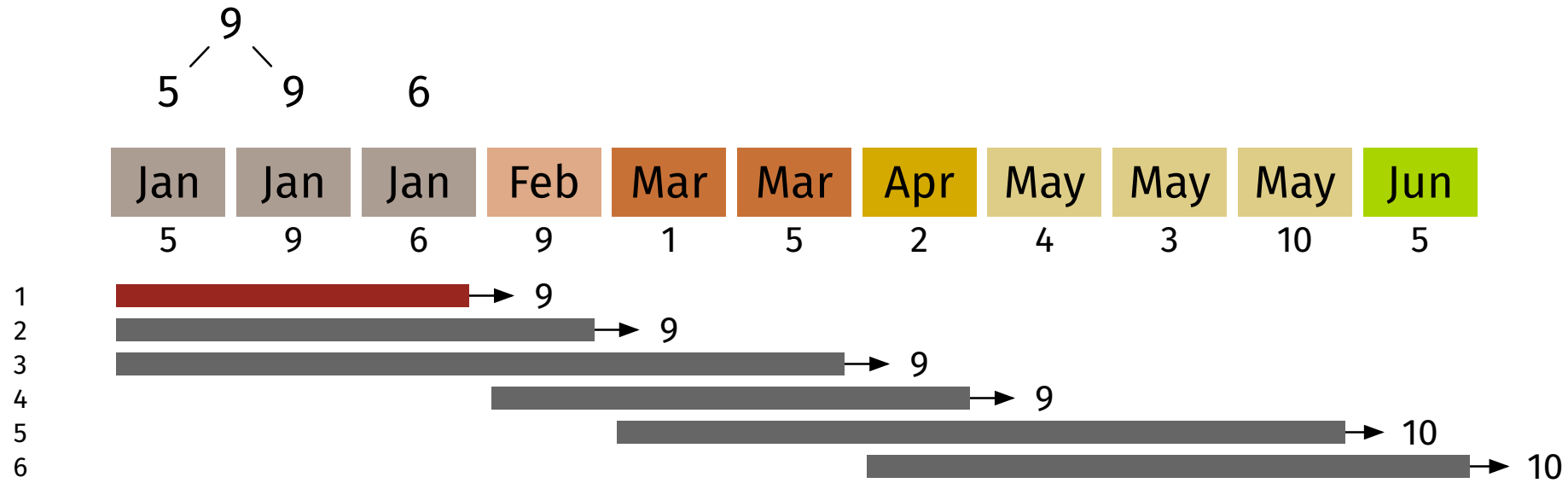
Ring Aggregates (Sum, Count, Average)

- Add new values: $O(1)$ per value
- Remove old values: $O(1)$ per value

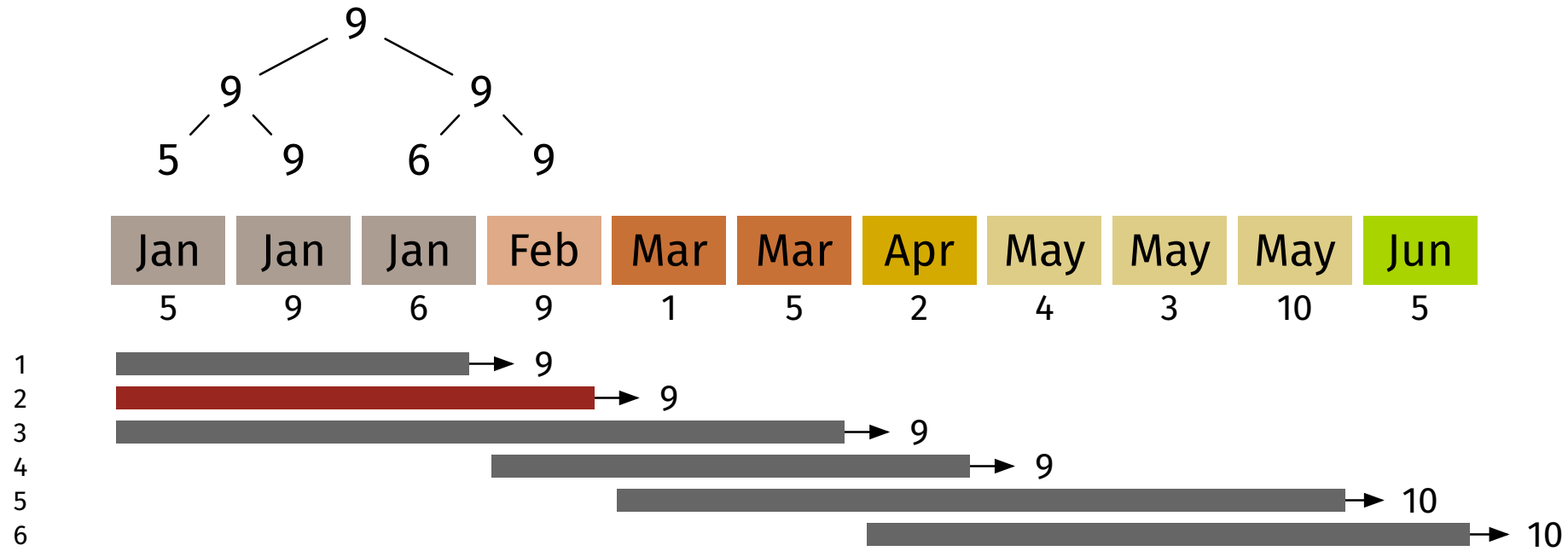
Semiring Aggregates (Min, Max)

- Rescan for new max: $O(|W|)$ per value

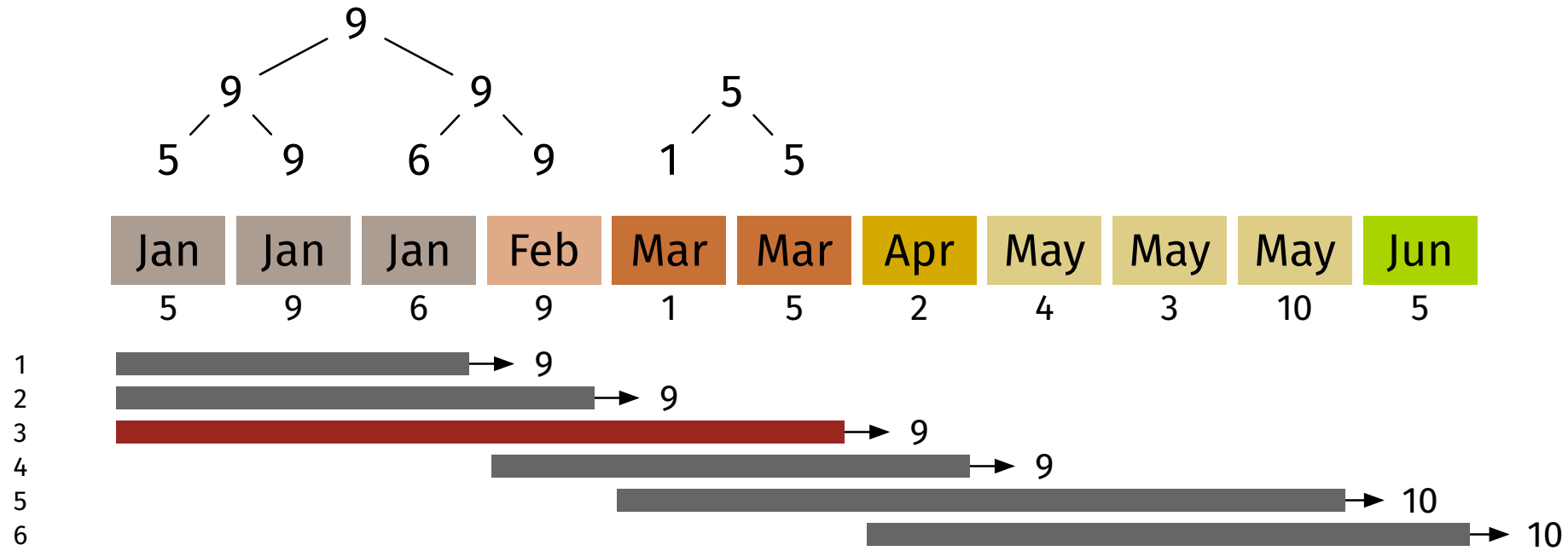
WINDOW Indexes



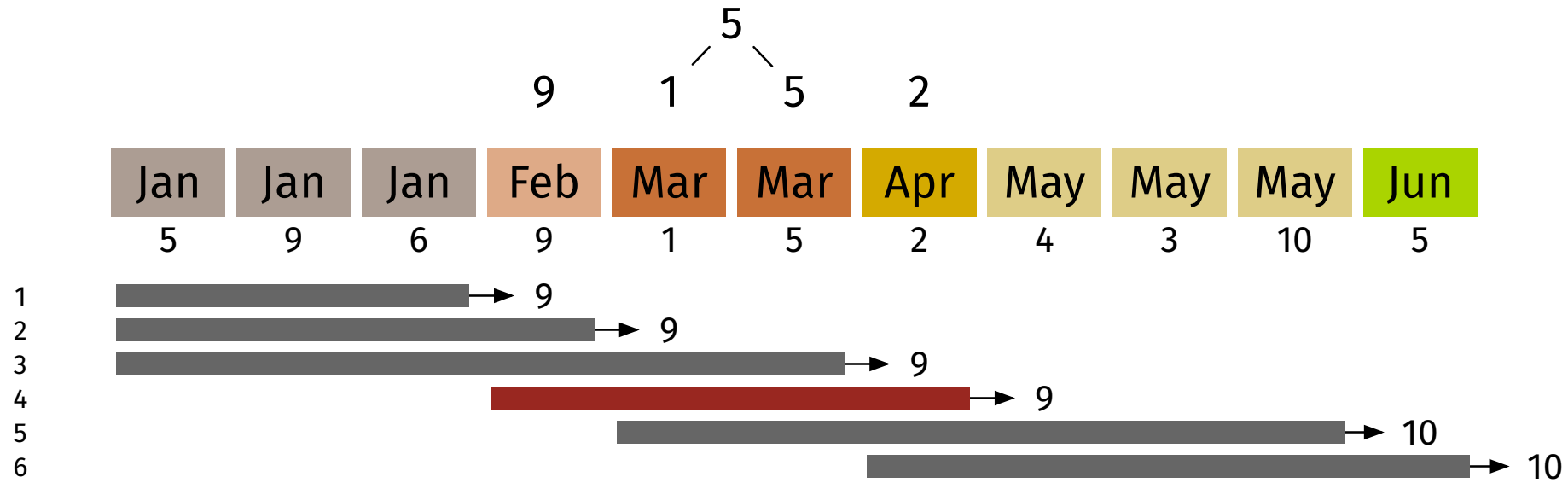
WINDOW Indexes



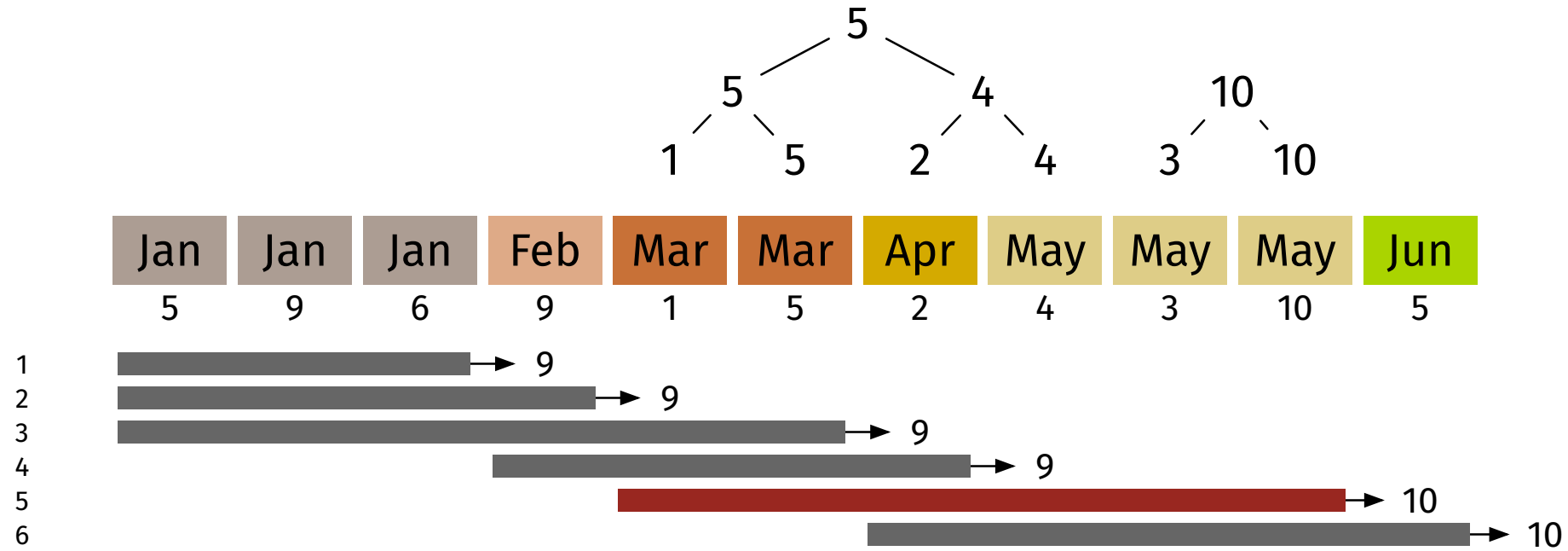
WINDOW Indexes



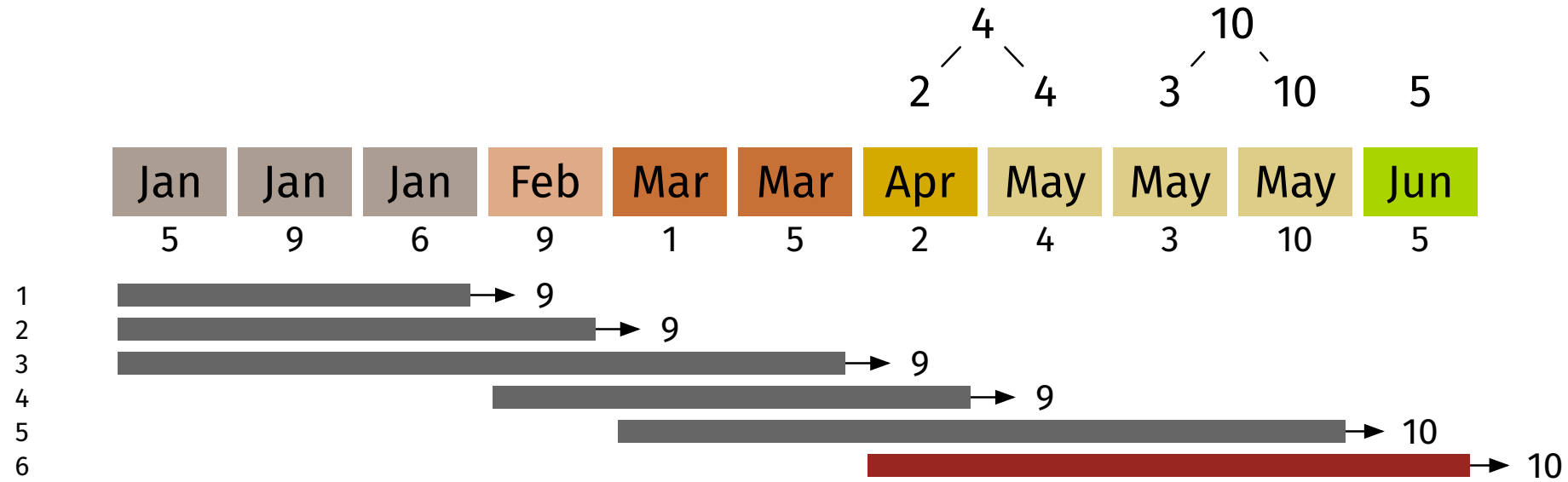
WINDOW Indexes



WINDOW Indexes



WINDOW Indexes



WINDOW queries

- Aggregate values over sequential ranges

Push vs Pull Data

- Push works better when sources produce data at different rates

Streaming Joins

- Streaming focuses on ripple-style joins

Streaming Indexing

- Linked Hash/Tree index for efficient windowed joins

Aggregation

- Sliding window aggregation