Data Stream Management Systems
Sliding Windows and CQL

Dr. Wenceslao PALMA
wenceslao.palma@ucv.cl

June 2011

Sliding Windows

Unbounded streams cannot be stored locally in a DSMS, and only the recent data items of a stream are usually of interest at any time. In general, this may be accomplished using a time-decay model, also referred to as an amnesic or fading model.

Time-decay model

A Time-decay model discounts each item in the stream by a scaling factor that is non-decreasing with time. **Exponential** and **polynomial** decay are two examples, as are **window models** where items within the window are given full consideration and items outside the window are ignored.

Time-decay models

A decay function takes some information about the ith item and returns a weight for this item. The weight of an item can be written as a function of its age a. The age a at time $t > t_i$ is simply $a = t - t_i$

- *Exponential.* $f(a) = exp(-\lambda a)$
- *Polinomial.* $f(a) = (a+1)^{\alpha}, \alpha > 0$
- *Sliding Windows.* Given a window size parameter W.

$$f_W(a) = \begin{cases} 1 & \text{if } a < W \\ 0 & \text{if } a \geq W \end{cases}$$

Sliding Windows

Windows classification

- *Direction of movement of the endpoints*. Two fixed endpoints define a **fixed window**, two sliding endpoints (either forward or backward, replacing old items as new items arrive) define a **sliding window**, and one fixed endpoint and one moving endpoint (forward or backward) define a **landmark window**.
- *Definition of window size*: Logical, or **time-based** windows are defined in terms of a time interval, whereas physical, (also known as count-based or **tuple-based**) windows are defined in terms of the number of tuples. Moreover, **partitioned windows** may be defined by splitting a sliding window into groups and defining a separate count-based window on each group.

Windows classification

- *Windows within windows.* In the **elastic window model**, the maximum window size is given, but queries may need to run over any smaller window within the boundaries of the maximum window. In the **n-of-N window model**, the maximum window size is N tuples or time units, but any smaller window of size n and with one endpoint in common with the larger window is also of interest.
- *Window update interval.* **Eager updating** advances the window upon arrival of each new tuple or expiration of an old tuple, but batch processing (**lazy updating**) induces a jumping window. Note that a count-based window may be updated periodically and a time-based window may be updated after some number of new tuples have arrived; these are referred to as mixed jumping windows. If the update interval is larger than the window size, then the result is a series of non-overlapping tumbling windows.

Continuous Query Language (CQL)

"The result of a continuous query at time T is the result of treating the streams up to T as relations and evaluating the query using standard relational semantics."

CQL

- CQL is an expressive SQL-based declarative language for registering continuous queries against streams and updatable relations.
- CQL is implemented in the STREAM DSMS.

CQL

- CQL is based on three classes of operators over streams and relations: relation-to-relation, stream-to-relation and relation-to-stream.
- In CQL a stream S is a possibly infinite bag of elements $\langle s, \tau \rangle$, where s is a tuple belonging to S and $\tau \in \mathcal{T}$ is the timestamp of the element.
- A relation R is a mapping from $\mathcal{T}$ to a finite but unbounded bag of tuples belonging to R.

CQL

- A **stream-to-relation** operator takes a stream S and generates a relation R with the same schema as S, this operator is based on the concept of sliding window.
- A **relation-to-relation** operator corresponds to standard relational algebraic operators, it takes one or more relations $R_1, ..., R_n$ as input and generates a relation R as output.
- A **relation-to-stream** operator takes a relation R as input and generates a stream S as output.

CQL

A window size in time units on a stream X contains a historical snapshot of a finite portion of the stream. An example query, computing a join of two timed-based windows of size 5 minutes each is shown below.

```
Select Distinct X.A
    From X[Range 5 min], Y[Range 5 min]
    Where X.B=Y.B
```

This query contains a stream-to-relation operator and a relation-to-relation operator that performs projection and duplicate elimination.

stream-to-relation

- Timed-based. In a time-based sliding window one stream S is specified by following the name of the stream with the Range keyword and a time interval enclosed in brackets. Ex.: *S [Range 30 seconds]*, *S [Now]*, *S [Range Unbounded]*
- Count-based. A tuple-based window defines its output relation over time by sliding a window of the last N tuples of an ordered stream. Count-based sliding windows may no tbe approriate when timestamps are not unique. Ex.: *S [Rows N]*, *S [Rows Unbounded]*.
- Partitioned windows. A partitioned window logically partitions a stream into different substreams based on equality of the attributes of data items computing a count-based sliding window independently on each substream. Ex.: *S [Partition by* A_1 *Rows 1]*.

Continuous Query Language (CQL)

relation-to-relation

An operator that performs projection and duplicate-elimination.

Select Distinct vehicleId
From PosSpeedStr [Range 2 min]

relation-to-stream

CQL has three relation-to-streams operators: *Istream*, *Dstream*, *Rstream*.

- $Istream(R) = \bigcup_{\tau \geq 0}((R(\tau) - R(\tau - 1)) \times \{\tau\})$. At each evaluation cycle *Istream* streams all new data items added to R.

  Select Istream(*)
  From PosSpeedStr [Range Unbounded]
  Where speed>65

- $Dstream(R) = \bigcup_{\tau > 0}((R(\tau - 1) - R(\tau)) \times \{\tau\})$. At each evaluation cycle *Dstream* streams all the data items removed from R.

- $Rstream(R) = \bigcup_{\tau > 0}(R(\tau) \times \{\tau\})$. At each evaluation cycle *Rstream* streams all the data items at once.

  Select Rstream(*)
  From PosSpeedStr [Now]
  Where speed>65

STREAM continuous query plans

Q_1:Select B, max(A)
From S1 [Rows 50000]
Group by B

Q_2:Select Istream(*)
From S1 [Rows 40000] , S2 [Range 600 seconds]
Where S1.A=S2.A