Semantics of Spatial Window over Spatio-Temporal Data Stream

Yi Yu[*,**], Talel Abdessalem[**]

[*]CIMS center, University of Tongji
Shanghai, China

[**]Ecole Nationale Supérieure des Télécommunications
LTCI - UMR CNRS 5141
46, rue Barrault, 75013 Paris - France
First.Last@enst.fr

Résumé. Dans les systèmes DSMS (Data Stream Management Systems), les données en entrée sont infinies et les requêtes sur celles-ci sont actives tout le temps. Dans le but de satisfaire ces caractéristiques, le fenêtrage temporel est largement utilisée pour convertir le flux infini de données sous forme de relations finies. Mais cette technique est inadaptée pour de nombreuses applications émergentes, en particulier les services de localisation. De nombreuses requêtes ne peuvent pas être traitées en utilisant le fenêtrage temporel, ou seraient traitées plus efficacement à l'aide d'un fenêtrage basé sur l'espace (fenêtrage spatial). Dans cet article, nous analysons la nécessité d'un fenêtrage spatial sur des flux de données spatio-temporels, et proposons, sur la base du langage de requêtes CQL (Continuous Query Language), une syntaxe et une sémantique associées au fenêtrage spatial.

1 Introduction

Data stream management systems (DSMS) have emerged to meet the needs of processing continuous changing, unbounded data and real-time responses. The applications include stock quoting, auction processing, network flow monitoring, moving objects monitoring [Abdessalem et al. (2007), Moreira et al. (2000)], etc. In these cases, the common features consist in : 1- the data sources are infinite and real-time changing, 2- queries over data have to produce continuous responses. To cope with the first feature, the window concept is proposed. The idea consists in transforming unbounded data stream into bounded data tables, then queries can be processed as in a traditional database system. For the second feature, query evaluation methods should be executed continuously resulting in a real-time changing of the response. As we mentioned above, window techniques are proposed for solving two issues in data stream processing : infinite data sources and continuous query. In current DSMS, the windowing operation is done using the timestamps of the input data (i.e. temporal attributes). For example, in a network traffic monitoring application it is not possible to store and analyze online the whole input data. We can just continuously monitor the situation for a bounded time interval,

for instance the latest 1 hour. In this case, we have an infinite stream of traffic data and a temporal window of 1 hour. The set of data belonging to the specified window is finite. Queries are then evaluated periodically (for instance, every 2 minutes) on the set of data belonging to the window. So, the result of the queries will change continuously (every 2 minutes). DSMS also propose windowed operators such as aggregate operators, join operator, select operator, etc. to enable answering complex queries. In addition to initial applications of data streams, the development of location-sensitive devices has also spurred the on-line location-aware services and real-time moving objects monitoring applications. Besides the common characteristics of DSMS, these applications further need to process spatial and spatio-temporal data for location services. In this case, the input data is supposed to be composed of at least two attributes : a spatial attribute and a timestamp. In the previous literatures, two main approaches are considered for continuous queries processing over spatio-temporal data streams. In the first approach [Patroumpas et Sellis (2004)], the windowing operation is done using only the timestamp attribute. Then, spatial restrictions are evaluated on each set of data composing the obtained windows "temporal windows". The problem with this approach is that the fixed order of operations(temporal/spatial restrictions) in a query evaluation plan can have a important impact on the performance of the system [Elmongui et al. (2006)]. In other terms, it may be inefficient in some cases when the temporal restriction is always done before the spatial one. In addition, many location services don't involve temporal attribute at all and only care about spatial information, e.g. Range or kNN query [Mokbel et Aref (2005)] etc. In the second approach, as proposed in the PLACE project [Mokbel et al. (2005)], the windowing operation is done using the spatial attribute. Then, temporal restrictions are evaluated on each set of data composing the obtained *Spatial Windows*.

To point out the necessity of spatial windows for spatio-temporal applications, we take the following example. Given the situation of a vehicle moving on the road. The vehicle sends the updates of its position periodically to the DSMS, and we would like to know its average speed in the last 50 kms. Temporal windowing is not efficient to answer this kind of query, because the size of the temporal window can't be decided in advance. The time spent by the observed object to cross the 50 kms may change significantly along the trajectory of the object.

A spatial windowing is more suitable to answer this kind of query. We may fall back on the spatial window (specified by the criteria "in the last 50 kms") to catch a finite data relation from the unbounded data stream. Only the semantics of the windowing operation changes because we use the spatial dimension instead of the temporal one. When processing the incoming streams, spatial window maintains its contents according to the spatial attribute of the input data (tuples). Only data within a certain area scope are of interest and are kept in the window.

In this paper, we focus on the extension of DSMS to the management of spatio-temporal data. We propose two kinds of spatial windows : a *static spatial window* and a *moving spatial window*. These operations are presented as a possible extension to the query language CQL (Continuous Query Language) [Arasu et al. (2006); Arasu et Widom (2004)]. Their semantics are defined based on the abstract semantics of CQL, and their expressiveness is demonstrated using examples of queries.

This paper is organized as follows. Section 2 presents the CQL query language. Section 3 describes our extension of CQL to support spatial windowing, and section 4 concludes the paper.

2 Preliminaries

In this section, we present the data model and the basic operations proposed in CQL. This query language was defined in the STREAM project [Arasu et al. (2006); Arasu et Widom (2004)]. Its syntax is close to the SQL-99 syntax and it is especially designed for continuous data processing over data streams. CQL assumes that the input tuples come in a time ordered sequence and the windowing operation is processed on the timestamp of each tuple. First, we present the basic functions and data domains, then we give the definitions of *stream* and *relation* and their related operators.

- T : Time domain ;
  T is a discrete, ordered time domain. A time instant is any value from T.
  Time attributes belong to T.
- TP : Tuple domain ;
  A *tuple* is a finite sequence of atomic values.
- Σ : Tuple multi-sets domain ;
  This is the domain of finite, but unbounded, bags of tuples.
- S : Stream domain ;
  This is the domain of multi-sets over $TP \times T$ (see Definition 2.1)
- R : Relation domain ;
  $R : T \rightarrow \Sigma$, this is the domain of functions that map time instants to bags of tuples (see Definition 2.2).
- $R2ROp : \Sigma \times ... \times \Sigma \rightarrow \Sigma$
  This the domain of functions that produce a bag of tuples from one or more bags of tuples. For example, the relational algebra operators (e.g. π, σ, etc.).
- $S2ROp : S \times T \rightarrow \Sigma$
  This is the domain of functions that converse a stream to bags of tuples. For example, the CQL operators *RANGE* and *SLIDE*.
- $R2SOp : \Sigma \times T \rightarrow S$
  This is the domain of functions that converse finite bags of tuples to a stream.
  Such functions in CQL are *IStream*, *DSream* and *RStream*.

Definition 2.1 :
A stream S is a (possibly infinite) bag (multi-sets) of elements $\langle s, t \rangle$, where s is a tuple belonging to the schema of S and $t \in T$ is the timestamp of the element.

Definition 2.2 :
A relation R is a mapping from each time instant $t \in T$ to a finite but unbounded bag of tuples, denoted *R(t)*, belonging to the schema of R.

CQL queries are composed from operators belonging to the three classes : Relation-to-Relation operators (from *R2ROp* domain), Stream-to-Relation operators (from *S2ROp* domain), and Relation-to-Stream operators (from *R2SOp* domain).

Let us rebuild the example given in the Introduction, and consider that we look for the average speed of a car in the past 60 minutes every 10 minutes. In this situation, sliding temporal

window can be used to extract tuples of latest 60 minutes with the sliding value of 10 minutes. By using CQL, query sentence can be expressed as follows :

Q_1 :

SELECT *RStream (AvgSpd(R))*
FROM *R=Stream_car* [*RANGE '60 minutes', SLIDE '10 minutes'*]
WHERE *R.ID=mycar*

This query is constructed from three classes of operators : *RANGE '60 minutes'* and *SLIDE '10 minutes'* respectively design the window size and moving step of the spatial window. They belong to the $S2ROp$ domain. A relational restriction operator restricts tuples to have the *ID* value equal to '*mycar*' after the conversion of stream to relation. An aggregate operator '*AvgSpd()*' calculates the average speed of '*mycar*' in the last 60 minutes by using tuples meeting the previous restriction. These two operators belong to the *R2ROp* domain. Finally, an *RStream* operator converts the content of result relation(object ID and average speed after previous calculation) to stream and transmits it to users. This operator belongs to the *R2SOp* domain. The three classes of operators in this example can be expressed by the semantics of CQL.

The semantics of CQL is specified using a meaning function $\mathcal{M}$ [Arasu et Widom (2004)]. The meaning function takes any query Q belonging to CQL and returns an "input-output" function $\mathcal{M}[\![Q]\!](r, s, t)$ after computation by Q. $\mathcal{M}[\![Q]\!](r, s, t)$ takes the streams and relations referenced in Q and a time instant t (e.g. *now*) as the input. Then it specifies the output produced by Q at time instant t.
Let us consisder the example of query Q_1 given above. In the first stage, a Stream-to-Relation conversion is done on the data stream $Stream\_car$. The meaning function of this operation is the following :

$$\mathcal{M}_{S2R}[\![Stream\_car\ [RANGE\ '60\ minutes',\ SLIDE\ '10\ minutes']]\!]$$
$$=$$
$$\lambda Stream\_car.\lambda t.\{tp : (tp, t') \in Stream\_car \wedge (t_{low} \leq t') \wedge (t' \leq t_{high})\},$$
$$\text{where } t_{high} = \lfloor t/10\ minutes \rfloor \times 10\ minutes, \text{ and } t_{low} = max\{t_{high} - 60\ minutes, 0\}$$

The expression $t_{high} = \lfloor t/10\ minutes \rfloor \times 10\ minutes$ computes the largest time instant multiple of 10 minutes and smaller then t. Intuitively, the Stream-to-Relation conversion defines its output tuples each *10 minutes*, only the tuples within the last *60 minutes* are contained in the output. Based on the obtained output relation, the evaluation of the *where* operation is done according the the following meaning function :

$$\mathcal{M}_{R2R}[\![Where\ R.ID{=}mycar]\!] = \lambda R.\lambda\ ID.\lambda mycar.\ \{\mathrm{r} : \mathrm{r} \in R \wedge r.ID = mycar\}$$

The meaning function of the operator *AvgSpd(R)* is :

$$\mathcal{M}_{R2R}[\![AvgSpd(R)]\!] = \lambda\ \mathrm{R}.\{\mathrm{Spd} : \mathrm{Spd{=}distance}/60\ minutes\}$$

In this case, *distance* represents the path that the monitored car has passed in the last 60 minutes. It is calculated using the position information that must contain each tuples in the window.

Finally, operator *RStream* of *R2SOp* domain converts its input relation to a stream. Its meaning function is :

$$\mathcal{M}_{R2S}[RStream(R)] \;=\; \lambda R.\lambda t\{\langle e, t'\rangle : t' \leq t \wedge e \in R(t)\}$$

3 Adding a spatial window operation to CQL

In this section, we propose an extension to CQL in order to support spatial windowing. We consider two categories of spatial windows : *stationary windows* and *moving windows*. The spatial coordinates of a stationary window do not change over time. However, the spatial coordinates of a moving window can change over time. To illustrate this, let's consider the following two spatial queries :

- Stationary Spatial Windows
  Suppose that we are interested in monitoring the trajectories of fishing ships in the east China sea. In this case, the interest area could be regarded as a stationary spatial window and the trajectory of each fishing ship consists in its continuous positions in that area.
- Moving Spatial Windows
  Recall the example of section 1 and expand it :"Query the average speed of a vehicle in the last 50 kms every time after it moves 10 kms". The spatial window here has the window size of 50 kms and the sliding step of 10 kms, which indicates that the spatial window is moving.

Based on this informal description of spatial windows, we extend in 3.1 the CQL data model and its basic operations presented in section 2. Then, we define in 3.2 the syntax and the semantics of the spatial window operation.

3.1 Extending the Data model

To be able to deal with spatial data, we add the following domains to CQL data model.

- G : Space domain ;
  This is the domain of spatial values. The spatial attributes of spatio-temporal data stream belong to this domain.
- $R = R_t \cup R_g$: Relation domain ;
  $R_g : G \rightarrow \Sigma$, this is the domain of functions that map spatial restriction to bags of tuples (see Definition 3.2).
  $R_t : T \rightarrow \Sigma$, this is the domain of functions that map temporal restriction to bags of tuples (this domain is denoted R in the CQL data model, see section 2).
- S : Stream domain ;
  This is the domain of multi-sets over $TP \times G \times T$ (see Definition 3.1)
- $S2ROp : S \times (G \cup T) \rightarrow R$;
  This is the domain of windowing functions that converse a spatio-temporal stream to bags of tuples. This will correspond to the new windowing operators *RANGE BY ...* [*RATTR SPACE* | *TIME*] and *SLIDE BY ...* [*SATTR SPACE* | *TIME*].
- $R2SOp : \Sigma \times G \times T \rightarrow S$;
  This is the domain of functions that converse finite bags of tuples to a spatio-temporal stream, by adding a spatial stamp and a temporal stamp to each tuple. This will correspond to the new CQL functions *IStream, RStream and DStream*.

A Spatio-temporal data stream consists of a stream of tuples, each one is stamped with a temporal attribute and a spatial attribute (i.e. each tuple have two stamps). Temporal windowing operators are executed on the basis of the temporal stamps, and spatial windowing operators are executed according to the spatial stamps. We define the stream and relation model in the case of spatio-temporal data as follows.

Definition 3.1 :
A stream S in the spatio-temporal case is a (possible infinite) bag of elements $\langle s, g, t \rangle$, where s is a tuple belonging to the schema of S, $g \in G$ is a spatial stamp, corresponding for example to the spatial coordinates of a moving object, and $t \in T$ is the temporal stamp of the element.

Definition 3.2 :
A relation *R(g,t)* is a mapping from each location $g \in G$ and each time instant $t \in T$ to a finite but unbounded bag of tuples, belonging to the schema of R. The content of a relation R changes over space and time.

In a data stream S, we denote by g the spatial stamp, and we denote by t the temporal stamp, of each tuple. These stamps are necessary for spatial and temporal windowing operations. Updates from different sources (for instance, moving objects) can flow in separate streams or may be incorporated into the same stream. When employing a spatial windowing operation, tuples will be filtered into bags of tuples according to their spatial stamps. When employing a temporal windowing operation, they will be filtered according to their temporal stamps. A continuous query on a spatio-temporal stream may be composed of only spatial or temporal windowing operations. It can also be composed at the same time of both temporal and spatial windowing operations.

3.2 Semantics and Syntax of Spatial Window

Since a data stream is infinite and the memory size is limited, the windowing approach is fundamental for the processing of continuous queries in DSMS [Patroumpas et Sellis (2006)]. In CQL syntax [Arasu et al. (2006); Arasu et Widom (2004)], the window operation is denoted by the keywords *RANGE*, *ROW* and *SLIDE*. In TelegraphCQ [Chandrasekaran et al. (2003)], the window operation is denoted by the expression *RANGE BY ... SLIDE BY* and in [Li et al. (2005); Maier et al. (2005)] the same CQL keywords *RANGE, SLIDE* are used to denote the window operation. In this paper, we use a syntax similar to TelegraphCQ to illustrate our spatial window operations.

The syntax we consider for the windowing operation is as follows :

$$RANGE\ BY\ \ v_1\ \ RATTR\ \ SPACE \mid TIME,\ \ SLIDE\ BY\ \ v_2\ \ SATTR\ SPACE \mid TIME$$

Where v_1 denotes the window size (range attribute) and v_2 denotes a step between two successive windows (slide attribute). The range and the slide attributes may be temporal or spatial values. This is indicated by the keywords *RATTR SPACE* and *RATTR TIME* for the range attribute, and by the keywords *SATTR SPACE* and *SATTR TIME* for the slide attribute. The range and the slide attribute may belong to the same domain (T or G) or not. In the following, we only consider the case where these two attributes are spatial values ($v_1 \in G$ and $v_2 \in G$).

3.2.1 Stationary Spatial Window

Let us take the example of the stationary spatial window given above at the beginning of Section 3. Since we only care about the trajectories of fishing ships in a certain region (east China sea), we can use a spatial window corresponding to the east China sea in order to only catch from the stream the tuples of interest. The trajectories consist of sets of position information. For a given object o_1, the following query gives its trajectory in the east China sea.

Q_2 :

SELECT *RStream(Stamps(*))*
FROM *Moving_Objects_Stream*
[*RANGE BY East_China_Sea RATTR SPACE*]
WHERE *ID*='o_1'

In this case, *Moving_Objects_Stream* denotes a spatio-temporal stream. The schema of this stream is composed of the *ID* attribute and some other attributes that indicate the speed and the orientation of the monitored moving objects. Function *Stamps(*)* returns the spatial and temporal stamps (g and t) of each tuple, which indicate the trajectory of the observed object. This query contains a *RANGE BY* value without a *SLIDE BY*. It means that the spatial window is stationary and do not change over time or space.

Formally, the semantics of the spatial window used in query Q_2 is specified by the following meaning function :

$$\mathcal{M}_{S2R}[\![Moving\_Objects\_Stream[RANGE\ BY\ \ East\_China\_Sea\ \ RATTR\ \ SPACE]]\!] = \lambda\, Moving\_Objects\_Stream.\ \lambda\, East\_China\_Sea. \{(tp, g, t) : (tp, g, t) \in Moving\_Objects\_Stream \ \wedge \ (g\ inside\ East\_China\_Sea)\}$$

While processing query Q_2, only the tuples within the spatial range *East_China_Sea* are preserved in the window. The content of the window is updated when a new tuple comes in. The fact that a tuple is qualified for the window or not is determined by the spatial operator *inside*, which determines if the spatial location g (spatial stamp) is inside the spatial area denoted *East_China_Sea*.

After the Stream-to-Relation operation (windowing operation), further processing on the window contents will be done. In this example, the *Where* clause and the *Stamps* operator will be performed. The meaning function of the where clause can be deduced easily from the semantics of CQL presented in section 2. The meaning function of the *Stamps* operator is as follows.

$$\mathcal{M}_{R2R}[\![Stamps(R)]\!] \ = \ \lambda R.\{(r.g, r.t) : r \in R\}$$

At the last stage, the result of query Q_2 is returned to the user in a stream format. This is done by the *RStream* operation. For each tuple added to its input relation, the *RStream* operator will re-evaluate completely its output and return all the tuples composing the output stream. Formally, the semantics of *RStream* is as follows :

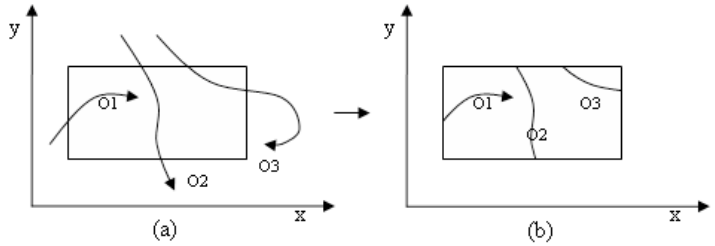

FIG. 1 – *Stream-to-Relation operation*

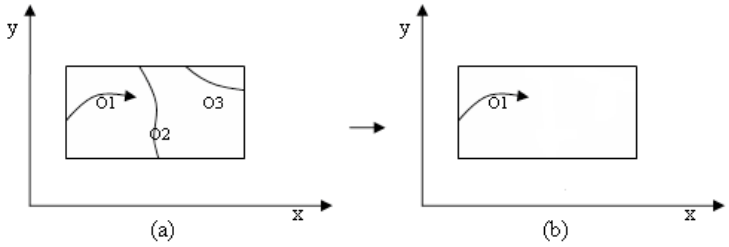

FIG. 2 – *Relation-to-Relation opertation for Window Content*

$$\mathcal{M}_{R2S}[\![RStream(R)]\!] = \lambda R.\{e : e \in R(g,t)\}$$

Figure 1 shows an example of static spatial window over a spatio-temporal stream (an *S2ROp*). Three moving objects (o_1, o_2 and o_3) are observed. Each new tuple represents an observation and contains the ID of the observed object, its velocity and its orientation. The temporal stamp of the tuple indicates the instant of the observation, and the spatial stamp indicates the location of the object at that instant. The trajectories shown in figure 1 represent the successive locations of the monitored objects. Figure 1.a represents the observations received up to now in the stream, and figure 1.b shows the subset of observations that are located inside the spatial window area *East_China_Sea*.

Figure 2 shows the result of the *R2ROp* filtering operation. The *Where* clause restricts the observations to only those corresponding to object o_1.

Figure 3 illustrates the conversion from relation to stream by *R2SOp*. In this example, *RStream* will output all the tuples of its input relation. In the case of an *IStream* operation, the result relation will be compared to the previous one and only the new tuples will compose the output.

object	g	t
O_1	(2,5)	T_1
O_1	(2.5,5.5)	T_2
O_1	(3,6)	T_3
O_1	…	…
O_1	(5,6)	T_n

→

T_1	T_2	T_3	…	T_n
O_1, (2,5)	O_1, (2.5,5.5)	O_1, (3,6)	O_1, …	O_1, (5,6)

FIG. 3 – *Relation-to-Stream opertation for Window Content*

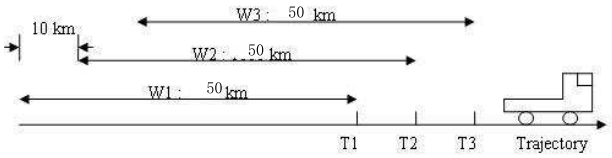

FIG. 4 – *Stream-to-Relation operation for Moving Window*

3.2.2 Moving Spatial Window

Let us take the example of the moving spatial window given at the beginning of Section 3. We add additional restrictions to this example, supposing that the car moves in a straight line and keeps the same direction. This query may be expressed as follows :

Q_3 :

SELECT *RStream (mycar,AvgSpd(R))*
FROM *R=Stream_mycar*
[*RANGE BY '50 km' RATTR SPACE, SLIDE BY '10 km' SATTR SPACE*]

The spatial windowing operation is done by the Stream-to-Relation operation *RANGE BY ... SLIDE BY*. The meaning function of this operation is as follows.

$$\mathcal{M}_{S2R}[\![Stream\_car\ [RANGE\ BY\ `50\ kms'\ RATTR\ SPACE,\ SLIDE\ BY\ `10\ kms\ SATTR\ SPACE']]\!]$$
$$=$$
$$\lambda Stream\_car.\lambda t.\ \{(tp, g, t') : (tp, g, t') \in Stream\_car \wedge g \leq d_{high} \wedge g \geq d_{low}\},$$
$$\text{where } d_{high} = \lfloor d_t / 10\ kms \rfloor \times 10\ kms,\ d_t = \sum_{i=1}^{t}(g_i - g_{i-1}),$$
$$\text{and } d_{low} = max\{d_{high} - 50\ kms, 0\}$$

The tuples in the relation are updated according to the parameter *RANGE BY* and *SLIDE BY*. Every time the monitored car moves 10 km ahead, the output result is re-evaluation and only the tuples having a spatial stamp g located inside the last *50 kms* will be preserved in the window.

Figure 4 illustrates the moving window operation in this example. The monitored car moves along a road. At time T_1, the car completed the first 50 kms and the tuples received in the stream up to T_1 compose the first window. At time instant T_2, the query window moves 10 kms ahead, and only the tuples corresponding to the last *50 kms* are kept in the window. Note that the time needed by the car to cross the "slide by" distance is not constant : $T_3 - T_2$ my be not equal to $T_2 - T_1$.

Similarly to query Q_1, the Relation-to-Relation operator *AvgSpd(R)* and the Relation-to-Stream operator *RStream* are used here to calculate the average speed and to output the the result in a stream format. Their meaning functions can be deduced easily from the example of Q_1.

4 Conlusion

The purpose of this paper was to define the semantics and general language syntax for a spatial windowing operation over data streams. Spatial windowing operation is useful for the querying of spatio-temporal data streams. Based on the continuous query language CQL, we proposed a syntax to express spatial windows and defined the semantics of this operation. this is the main contribution of this paper. Next steps will be the implementation of the windowing operation presented in this paper, its performance analysis on a real world application, and the analysis of more complex spatial windowing cases.

Références

Abdessalem, T., R. Chiky, G. Hébrail, et J. L. Vitti (2007). Traitement de données de consommation électrique par un système de gestion de flux de données. In M. Noirhomme-Fraiture et G. Venturini (Eds.), *Actes des cinquièmes journées Extraction et Gestion des Connaissances (EGC'07), Namur, Belgique, 23-26 janvier 2007*, Volume RNTI-E-9 of *Revue des Nouvelles Technologies de l'Information*, pp. 521–532. Cépaduès-Éditions.

Arasu, A., S. Babu, et J. Widom (2006). The CQL continuous query language : semantic foundations and query execution. *VLDB Journal 15*(2), 121–142.

Arasu, A. et J. Widom (2004). A denotational semantics for continuous queries over streams and relations. *SIGMOD Record 33*(3), 6–11.

Chandrasekaran, S., O. Cooper, A. Deshpande, M. J. Franklin, J. M. Hellerstein, W. Hong, S. Krishnamurthy, S. Madden, V. Raman, F. Reiss, et M. A. Shah (2003). TelegraphCQ : Continuous dataflow processing for an uncertain world. In *The Conference on Innovative Data Systems Research (CIDR'03)*.

Elmongui, H. G., M. Ouzzani, et W. G. Aref (2006). Challenges in spatiotemporal stream query optimization. In P. K. Chrysanthis, C. S. Jensen, V. Kumar, et A. Labrinidis (Eds.), *Fifth ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE'2006), June 25, 2006, Chicago, IL, USA, Proceedings*, pp. 27–34.

Li, J., D. Maier, K. Tufte, V. Papadimos, et P. A. Tucker (2005). Semantics and evaluation techniques for window aggregates in data streams. In F. Özcan (Ed.), *Proceedings of the ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland, USA, June 14-16, 2005*, pp. 311–322.

Maier, Li, Tucker, Tufte, et Papadimos (2005). Semantics of data streams and operators. In *the 10th International Conference on Database Theory (ICDT'05)*.

Mokbel, M. F. et W. G. Aref (2005). GPAC : generic and progressive processing of mobile queries over mobile data. In P. K. Chrysanthis et G. Samaras (Eds.), *6th International Conference on Mobile Data Management (MDM'2005), Ayia Napa, Cyprus, May 9-13, 2005*, pp. 155–163.

Mokbel, M. F., X. Xiong, M. A. Hammad, et W. G. Aref (2005). Continuous query processing of spatio-temporal data streams in PLACE. *GeoInformatica 9*(4), 343–365.

Moreira, J., C. Ribeiro, et T. Abdessalem (2000). Query operations for moving objects database systems. In K.-J. Li, K. Makki, N. Pissinou, et S. Ravada (Eds.), *Proceedings of the*

Eighth ACM Symposium on Advances in Geographic Information Systems (ACM-GIS'00), November 10-11, 2000, Washington D.C., USA, pp. 108–114.

Patroumpas, K. et T. K. Sellis (2004). Managing trajectories of moving objects as data streams. In J. Sander et M. A. Nascimento (Eds.), *2nd International Workshop on Spatio-Temporal Database Management (STDBM'04), Toronto, Canada, August 30, 2004*, pp. 41–48.

Patroumpas, K. et T. K. Sellis (2006). Window specification over data streams. In T. Grust, H. Höpfner, A. Illarramendi, S. Jablonski, M. Mesiti, S. Müller, P.-L. Patranjan, K.-U. Sattler, M. Spiliopoulou, et J. Wijsen (Eds.), *EDBT Workshops*, Volume 4254 of *Lecture Notes in Computer Science*, pp. 445–464. Springer.

Summary

In DSMS (Data Stream Management Systems), input data is infinite and the queries over it are active all the time. In order to cope with these features, Time-based window is widely used to convert infinite data stream into bounded relations. But, this technique is not adequate for many emerging applications, especially for location service applications. Many queries can not be processed by the Time-based windows or may be more efficient using Space-based windows (Spatial windows). In this paper, we analyze the necessity of a spatial windowing over spatio-temporal data streams and, based on the DSMS query language CQL (Continuous Query Language), we propose a syntax and semantics for spatial windows.