

Implementing and Evaluating Snapshots + Events Spatiotemporal Modelling Approach

Cristian Vidal

Universidad de Talca
Ingeniería Informática Empresarial
cvidal@utalca.cl

Akbar Ghobakhlou

Auckland University of Technology
Geoinformatics Research Centre
akbar@aut.ac.nz

Sara Zandi

Auckland University of Technology
Geoinformatics Research Centre
szandi@aut.ac.nz

ABSTRACT

Traditional approaches for modelling spatiotemporal information (snapshots, states and events) are not very efficient and usually not capable of retrieving information based on specific spatiotemporal query. The snapshots modelling technique is the oldest and simplest approach and does not support any spatiotemporal query. The most current approach is the events modelling technique which allows data retrieval using spatiotemporal query. However, deductive capacities are needed for developing a system which is not usually present in traditional databases. One idea is to retain the advantages of each individual traditional approach while combining them to build and develop an efficient spatiotemporal information system.

A case study is presented for building a spatiotemporal information system combining snapshots and events approaches which overcomes some of the problems associated with snapshots and events approaches on their own. This work examine the hybrid (snapshots + events) spatiotemporal modeling technique with generic implementation details.

Keywords

Snapshots, states, events, (snapshots + events), spatiotemporal modelling approach.

1. INTRODUCTION

There has been considerable research effort in the last decade directed to incorporating time and space in spatiotemporal modelling. The spatiotemporal data model is the key to handling spatial and temporal data simultaneously [15]. The inherent characteristics associated with changes in time and space is beneficial to represent real world geographical phenomena. There are several classical approaches for spatiotemporal information modelling such as snapshots, states and events [2, 3, 4, and 5]. A new spatiotemporal approach based on snapshots and events was proposed in 2005 [6]. This paper describes an implementation of the snapshot and event modelling approach based on [6] and evaluates performances for number of spatial and temporal queries. Spatiotemporal information systems have been very useful for some special subjects such as to have a record of some existing geographic zones and to show some consequences in

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

splitting or merging these. In Chile for example, two new geographic zones appeared for splitting some existing zones in 2007 [1].

This paper is structured as follows. Section 2 presents the classical spatiotemporal information modelling approaches. Section 3 describes the snapshots + events approach and section 4 present a case study. Section 5 explain database implementation in details and section 6 presents evaluation of number of classical spatiotemporal queries. Conclusions and future works are present in final section.

2. CLASSICAL SPATIOTEMPORAL MODELLING APPROACHES

The first approach to modelling spatiotemporal information is based on snapshots of the objects in the system. This model was originally proposed in [2] and allows the representation of a collection of temporarily homogeneous units in a particular subject. In a more simple form, snapshots are static models or images of the World State at particular time instants (photographs of the objects under study). Figure 1 illustrates the snapshots modelling technique. A set of snapshots constructed in time T_i where each cell or sub-region corresponds to a particular snapshot item. In this manner snapshots model can be seen as a sample dynamic phenomenon in a sequence of time instants [5]. This view of the snapshots modelling approach causes a series of practical problems that are analysed below.

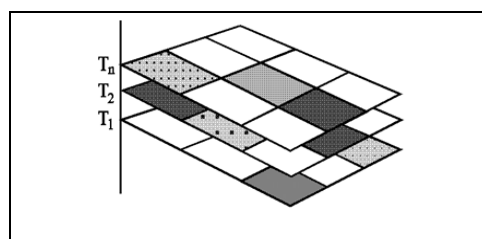


Figure 1: Snapshots modelling technique adopted from[2].

The problem with snapshots approach is that it cannot adequately provide information related to the time intervals between successive snapshots for spatiotemporal queries. In other words, this approach unable to retrieve the time of changes when comparing successive snapshots or photos for an object in question [2, 5].

2.1 States Approach

The model based on states introduces the notion of timestamp for all objects. In such model the idea is to have a record of all the

temporary object states. The state of an object is determined by the value of its attributes. Thus, an object undergoes a state change when the values of its attributes changes.

The value of attribute is recorded as state of an object changes. For example, if we define a class “**Region**” with attributes “**Attribute**” and “**State**” then any change in state associated with class Region is recorded as shown in Tables 1 and 2.

Table 1: Example of the classical state-based model for Chile’s Central Region

Name	Attribute	State	Vs
Central Region	Population	1000 – 10000	01/01/1810
Central Region	Population		31/12/1899
Central Region	Population	10000 – 30000	01/01/1900
Central Region	Population		31/12/1949
Central Region	Population	30000 –	01/01/1950

Table 2: Example of the between attribute in state-based model for Chile’s Central Region

Name	Population	Area	Vs – Ve
Central Region	1000 – 10000	7000 m ²	01/01/1810 – 31/12/1899
Central Region	10000 – 30000	3000 m ²	01/01/1900 – 31/12/1949
Central Region	30000 –	2000 m ²	01/01/1950 – NOW
Central Region	Area	2000 m ²	01/01/1950
Central Region	Area		31/12/1949
Central Region	Area	3000 m ²	01/01/1900
Central Region	Area		31/12/1899
Central Region	Area	7000 m ²	01/01/1810

This model widely used in practical developments of temporal databases uses a timestamp based on intervals of time, transaction time and valid time (Ts-Te and Vs-Ve, respectively) [3, 17]. Those timestamps can be associated with any relationship where the objects are involved. The NOW value at the end of a time interval represents a valid time interval open at its right edge (or currently valid) [5].

The states spatiotemporal modelling approach is able addresses the problem with the snapshot approach. However, discard useful information related to the phenomena that causes the state changes.

2.2 Events Approach

A model based in events allows recording spatiotemporal event occurrences and to derive the object’s temporal state. An event-based modelling approach allows modelling participation and involvement relationships between objects and events [11, 12]. A

model based on events is a complete model because it reflects all occurrences of spatiotemporal events.

This model type usually restricts the set of events to information explicit in the problem domain. This way the loading of unnecessary and irrelevant information is avoided, which in theory reduces the otherwise high computational cost of data transfer and object association [14]. Although this model enables improved spatiotemporal query results, it is achieved through a computationally expensive process. This is mainly due to the absence of the object’s state in the model [5].

The literature review emphasizes the following types of events associated with spatial objects:

- *Create*: An event that creates an object.
- *Change*: An event that changes a property of an object without changing the identity of the object. It is distinguished between change of attributes (internal change) and changes of membership (external change).
- *Destroy*: An event that destroys an object.
- *Split*: An event that creates one or more new objects from an existing object, which undergoes a change in their spatial boundaries (and possibly some non-spatial attributes) caused by the occurrence of this event.
- *Merge*: An event that may result in a new object creation, through the merger of two or more objects that no longer exist, or a change in the boundaries of an existing object (and possibly some non-spatial attributes) due to merge it with one or more objects that no longer exist.

3. SNAPSHOTS + EVENTS APPROACH

The snapshots + events approach combines snapshot and event approaches to model spatiotemporal information. This method enables processing both time interval and event queries simultaneously [6]. This approach is an application example of the Shannon – Nyquist’s sample theorem where the sample period has a big effect over the system [5].

This method generates successive snapshots when changes occurs over objects and keep a log of the events which cause these changes . Figure 2 depicts the main ideas of the snapshots + events approach.

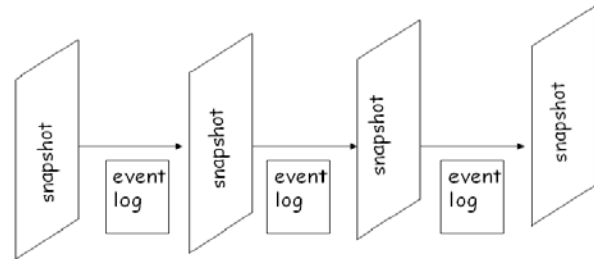


Figure 2: Snapshots + Event based approach adopted from[6]

Next section it will be showed a study case for applying and evaluating some implementation values of this approach (answering capacity, necessary time for answering some spatiotemporal queries, implementation possibilities with traditional software and database systems).

4. CASE STUDY

This case study designed to illustrate an implementation of the snapshots + events approach described in Section 3. It consists a set of objects O which are moving within a set of spatial regions R at certain time. Spatial regions are defined as a partition in space where:

$$\forall r_1, r_2 \in R \mid r_1 \neq r_2 \bullet r_1 \cap r_2 = \emptyset \text{ and } \bigcup_{r_i \in R} r_i = R$$

Figure 3 illustrate a simplified spatial representation including four regions R1, R2, R3 and R4 .

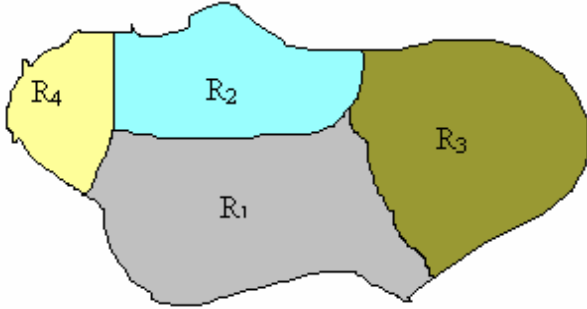


Figure 3: An exemplified and simplified spatial representation, including four regions R1, R2, R3 and R4

The location of spatiotemporal objects can be retrieved from the occurrences of events in the case study. There are two event classes defined: appearing and disappearing. The appearing event instances are identified with the 1010 code, while the disappearing event instances are classified with the 1020 code. The objects and regions are identified by their unique codes.

The data for this case study was generated through a moving objects simulator considering all the possible events (i.e. disappearing and appearing objects). Table 3 shows the table structure and sample data used.

Table 1. A part of a file example of the study case

Id_Region	Time	Event_Type	Object_Id
80	0.345000	1020	8687
81	0.345000	1010	8687
10	0.345000	1020	6185
11	0.345000	1010	6185
71	0.345000	1020	7965
.	.	.	.
.	.	.	.
49	0.535000	1010	6891
99	0.535000	1020	8661
89	0.535000	1010	8661
19	0.535000	1020	6569
9	0.535000	1010	6569
79	0.535000	1020	3397

The following entities are included in the case study:

- Number of region: 100
- Number of objects: 10000
- Number of event occurrences: 115824
- Beginning time: 0
- Ending time: 1

5. IMPLEMENTATIONS

This section describes implementation details of the spatiotemporal approach in Oracle 10g Express Edition [7]. Data generated by the event simulator stored in a table called “Temp” as shown in Figure 4.

```
CREATE TABLE Temp (
  Id_Event NUMBER,
  Time_T NUMBER,
  Id_Region NUMBER,
  Type_Object NUMBER,
  Id_Object NUMBER);
```

Figure 4: Table Temp SQL syntax

5.1 Snapshots + Events Implementation

A data model was created with integrations of spatiotemporal event occurrences and snapshots generated based on a given time granularity. This is to avoid the overloading problems associated with the traditional event modelling approach.

This modelling employs the SEST-Index method for spatiotemporal modelling proposed in [6]. As illustrated in Figure 5, the state of the objects in the snapshot t_0 are stored in R_0 , and the events that modify the geometry of objects in the temporal interval $(t_0; t_1)$, are stored in $\log L_0$. Therefore, to recover the state of the database at an instant t with $t_0 < t < t_1$, we start from the R-tree in instant t_0 and update objects' attributes (i.e., location) with the information of $\log L_0$. The entire index is a data structure A which is a sequence of snapshots A_i . This implementation reduces the number of records to be searched in order to answer certain queries comparing with the traditional events modelling approach.

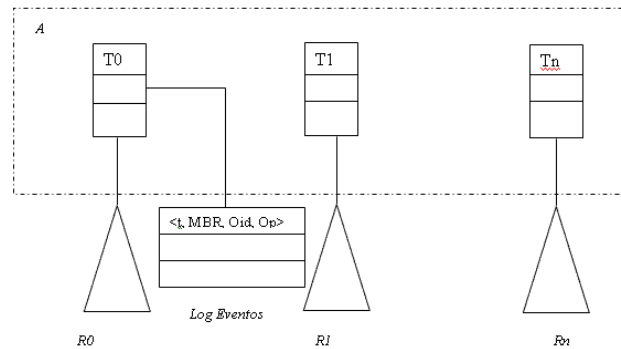


Figure 5: Spatiotemporal modelling approach based in snapshots + events adopted from[6].

Figure 6 illustrates the implemented conceptual model where each snapshot is associated explicitly with a *Time* class.

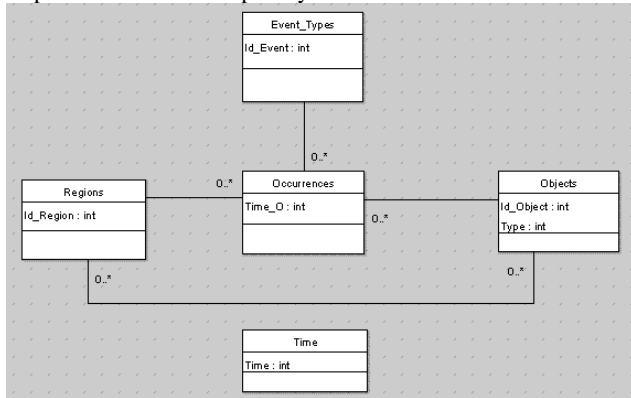


Figure 6: UML class diagram of snapshots + events modelling approach

The implementation of snapshots + events spatiotemporal modelling was realised through the *Occurrences* and *Snapshots* and *Times* tables. Table *Times* stores the time instants that the snapshots were created. The SQL code for creating the mentioned tables is shown in Figure 7.

```

CREATE TYPE Object_ AS OBJECT (Id NUMBER, Type NUMBER);
CREATE TYPE Region AS OBJECT (Id NUMBER);
CREATE TYPE Event_Type AS OBJECT (Id NUMBER);
CREATE TABLE Event_Types OF Event_Type;
CREATE TABLE Objects OF Object_;
CREATE TABLE Regions OF Region;
CREATE TABLE Occurrences (Time_O NUMBER,
Event_TypeRef REF Event_Type SCOPE IS Event_Types, ObjectRef
REF Object_ SCOPE IS Objects, RegionRef REF Region SCOPE IS
Regions);
CREATE TABLE Snapshots (Time_S NUMBER, ObjectRef REF Object
SCOPE IS Objects, RegionRef REF Region SCOPE IS Regions);
CREATE TABLE Times (Time_G NUMBER);
  
```

Figure 7: SQL code for creating tables required for the model

Store procedure *Snapshots_Events_Generate* (see Figure 8 and Figure 9) uses the “Temp” table for generating the associated records for each table in the model. This procedure takes a “Delta” value as an input argument for defining the time to store the snapshots and a “Final” input argument for defining the ending of the generating snapshots cycle. For example, for a “Delta” value of 10 and a “Final” value of 200, the snapshots is stored for the times [0, 5, 10, 15, ..., 200]. An events log is stored between snapshots.

```

CREATE OR REPLACE PROCEDURE
Snapshots_Events_Generate(Delta IN NUMBER, Final IN
NUMBER, Salida OUT NUMBER) IS
    CURSOR Temp_Cursor (T NUMBER) IS SELECT * FROM Temp A
WHERE A.Id_Event=1010 AND A.Time_T <= T AND
(SELECT COUNT(*) FROM Temp B WHERE B.Id_Event=1020
AND B.Id_Region=A.Id_Region AND B.Id_Object=A.Id_Object
AND B.Time_T > A.Time_T AND B.Time_T <= T)=0
ORDER BY A.Time_T ASC;

    CURSOR Temp_Cursor2 (T1 NUMBER,T2 NUMBER) IS SELECT
* FROM Temp WHERE Time_T > T1 AND Time_T <= T2 ORDER
BY Time_T ASC;
    Row_Temp Temp%ROWTYPE;
    Count_ NUMBER;
    RegionRef REF Region;
    ObjectRef REF Object_;
    Event_TypeRef REF Event_Type;
    TimeAct NUMBER;
BEGIN
    TimeAct:=0;
    Salida:=0;

    WHILE TimeAct<=Final LOOP
        INSERT INTO Times VALUES(TimeAct);
        OPEN Temp_Cursor(TimeAct);
        LOOP
            FETCH Temp_Cursor INTO Row_Temp;
            EXIT WHEN Temp_Cursor%NOTFOUND;
            /*Adding Object*/
            SELECT COUNT(*) INTO Count_ FROM Objects WHERE
            Id=Row_Temp.Id_Object;
            IF (Count_=0) THEN
                INSERT INTO Objects VALUES(Row_Temp.Id_Object,0);
            END IF;
            /*Adding Region*/
            SELECT COUNT(*) INTO Count_ FROM Regions WHERE
            Id=Row_Temp.Id_Region;
            IF (Count_=0) THEN
                INSERT INTO Regions VALUES(Row_Temp.Id_Region);
            END IF;
            /*Adding Snapshot*/
            SELECT REF(r) INTO RegionRef FROM Regions r WHERE
            r.id=Row_Temp.Id_Region;
            SELECT REF(o) INTO ObjectRef FROM Objects o WHERE
            o.id=Row_Temp.Id_Object;
            INSERT INTO Snapshots VALUES(TimeAct, ObjectRef,
            RegionRef);
        END LOOP;
    CLOSE Temp_Cursor;
  
```

Figure 8: SQL code of Snapshots_Events_Generate store procedure (I).

```

/*Events Log*/
OPEN Temp_Cursor2(TimeAct, TimeAct + Delta);
LOOP
  FETCH Temp_Cursor2 INTO Row_Temp;
  EXIT WHEN Temp_Cursor2%NOTFOUND;

  /*Adding Object*/
  SELECT COUNT(*) INTO Count_ FROM Objects WHERE
  Id=Row_Temp.Id_Object;
  IF (Count_ =0) THEN
    INSERT INTO Objects VALUES(Row_Temp.Id_Object,0);
  END IF;

  /*Adding Region*/
  SELECT COUNT(*) INTO Count_ FROM Regions WHERE
  Id=Row_Temp.Id_Region;
  IF (Count_ =0) THEN
    INSERT INTO Regions VALUES(Row_Temp.Id_Region);
  END IF;

  /*Adding EventType*/
  SELECT COUNT(*) INTO Count_ FROM Event_Types
  WHERE Id=Row_Temp.Id_Event;
  IF (Count_ =0) THEN
    INSERT INTO Event_Types
  VALUES(Row_Temp.Id_Event);
  END IF;

  /*Adding Event Occurrence*/
  SELECT REF(r) INTO RegionRef FROM Regions r WHERE
  r.Id=Row_Temp.Id_Region;
  SELECT REF(o) INTO ObjectRef FROM Objects o WHERE
  o.Id=Row_Temp.Id_Object;
  SELECT REF(e) INTO EventTypeRef FROM Event_Types e
  WHERE e.Id=Row_Temp.Id_Event;

  INSERT INTO Occurrences VALUES(Row_Temp.Time_T,
  EventTypeRef, ObjectRef, RegionRef);
  END LOOP;

  CLOSE Temp_Cursor2;

  TimeAct:=TimeAct + Delta;
  END LOOP;

END Snapshots_Events_Generate;

```

Figure 9: SQL code of Snapshots_Events_Generate store procedure (II).

6. EVALUATIONS

In this section there are presented three basic spatio-temporal queries to show that (Snapshots + Events) can work with them.

Question 1: Which region the object X is located at the time T?

The primary advantage of this modelling approach is that it always reduces the number of records that falls within the time intervals T. Figure 10 shows the algorithm and Figures 11, 12 shows its implementation as Oracle SQL. Figure 13 shows an example of the function work.

1. It is located the time nearest to the snapshots generating time (**T_Snapshot**), lower than the time **T**.
2. It is established the existence or not of a snapshot over the object **X** in the time **T_Snapshot**.
3. It is asking if the query time (**T**) is the same as the snapshot generating time (**T_Snapshot**).
T = T_Snapshot.

If they have the same value (**T = T_Snapshot**) and there is a snapshot for the object X, then the snapshot associated region identity (Id Region) is returned. If the time values are the same, but there is not a snapshot for the object X, then -1 is returned.

If the time values are not the same (**T <> T_Snapshot**) and there is a snapshot for the object X in T_Snapshot, then:

- It is asked if occurred a disappearing event (1020 event) over the object X in a time greater than T_Snapshot and lower or at the same time that the time T.
 - If that event had place, then it is asking if the object X appeared again between **T_Snapshot** and **T** (1010 event) and is still visible. If it is like that, the associated region identity (Id Region) is gotten. In opposite case -1 is returned (invisible Object).
- If a disappearing event haven't been occurred then the object snapshot is still valid. It is gotten the Identity of the associated snapshot region in T_Snapshot time.
- If the times are not the same and there is not an associated snapshot for the object X en the T_Snapshot time, then it is asking if the object appeared between the T_Snapshot and the T times (1010 event) and it is still visible. If it is like that, it is gotten the associated Region Id. It is returned -1 in opposite case.

Figure 10: Algorithm related to the question 1

```

CREATE OR REPLACE FUNCTION Object_Region(X NUMBER,T
NUMBER) RETURN NUMBER IS
  T_Snapshot NUMBER;
  Id_Region NUMBER;
  Count_ NUMBER;
BEGIN
  Id_Region:=-1;
  /*Select Latest Snapshot */
  SELECT MAX(Time_G) INTO T_Snapshot FROM Times WHERE
Time_G <= T;
  /* Are there any Snapshot for the Object at the time
T_Snapshot?*/
  SELECT COUNT(*) INTO Count_ FROM Snapshots S WHERE
S.ObjectRef.id = X AND S.Time_S = T_Snapshot;
  IF (T = T_Snapshot AND Count_ > 0) THEN
    SELECT S.RegionRef.Id_Region INTO Id_Region FROM
Snapshots S WHERE S.ObjectRef.Id_Region = X
AND S.Time_S = T;
  /* If there is any Snapshot? (Is visible the Object?)*/*
  ELSIF (Count_ > 0) THEN
    /*Is the Object disappeared?*/
    SELECT COUNT(*) INTO Count_ FROM Occurrences O
WHERE O.EventTypeRef.id_Event =
1020 AND O.ObjectRef.id_Object = X AND O.Time_O >
T_Snapshot AND O.Time_O <= T;
    IF (Count_ > 0) THEN
      SELECT O.RegionRef.Id_Region INTO Id_Region FROM
Occurrences O WHERE
      O.EventTypeRef.Id_Event = 1010 AND
O.ObjectRef.Id_Object = X AND O.Time_O > T_Snapshot AND
O.Time_O <= T AND (SELECT COUNT(*) FROM
Occurrences OO WHERE
OO.EventTypeRef.Id_Event = 1020 AND
OO.ObjectRef.Id_Object = X AND OO.Time_O > O.Time_O AND
OO.Time_O <= T) = 0;
    ELSE
      SELECT S.RegionRef.Id_Region INTO Id_Region FROM
Snapshots S WHERE S.ObjectRef.Id_Object =
X AND S.Time_S = T;
    END IF;
  ELSE /* If no Snapshot*/
    SELECT O.RegionRef.Id_Region INTO Id_Region FROM
Occurrences O WHERE O.EventTypeRef.Id_Event = 1010 AND
O.ObjectRef.Id_Object = X AND O.Time_O > T_Snapshot AND
O.Time_O <= T AND (SELECT COUNT(*) FROM
Occurrences OO WHERE OO.EventTypeRef.Id_Event = 1020
AND OO.ObjectRef.Id_Object = X AND OO.Time_O > O.Time_O
AND OO.Time_O <= T) = 0;
    END IF;
  RETURN Id_Region;

```

Figure 11: Implementation of SQL function for the question 1 (I)

```

EXCEPTION
  WHEN NO_DATA_FOUND
  THEN RETURN Id_Region;
END Object_Region;

```

Figure 12: Implementation of SQL function for the question 1 (II)

```

declare Object_number; Time_number; Region_number;
begin Object_:=8687; Time_:=0.345;

Region_:=object_region(Object_ , Time_);

dbms_output.put_line(Region_);
end;

-----

81

Sentencia procesada.

0,04 segundos

```

Figure 13: Implementation of SQL function for the question 1 (III)

Question 2: Which region the event E occurred at the time T?

This is an easy task for this model to retrieve adequate result since this model has stored the events occurrence time. Figure 14 is the SQL code to perform this query.

```

SELECT O.RegionRef.Id_Region FROM Occurrences O WHERE
O.EventTypeRef.Id_Event = E AND O.Time_O = T

```

Figure 14: SQL code to perform query in the question 2.

Question 3: When was the object O in the region R?

With the event modelling approach, the object's states can be derived from events recording. First it is necessary to localize an appearing event for the object O in the region R, with an occurrence at time T_1 . It is necessary to localize the next disappearing event for the object O in the region R at the time occurrence T_2 where $T_2 > T_1$. In other words, the state of object O remains the same in the region R between the interval $[T_1, T_2]$ (see Figure 15). The SQL syntax is shown in Figure 16.

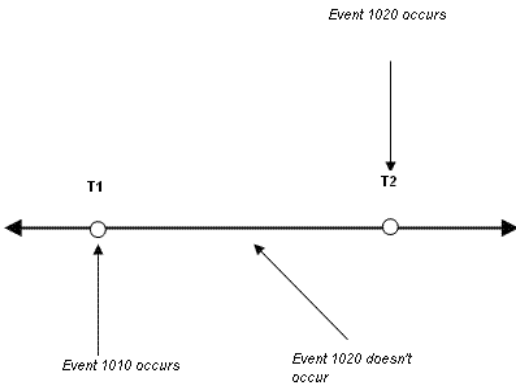


Figure 15: An object spatial location in a time interval

```

SELECT O1.Time_O, NVL(O2.Time_O, NOW) FROM Occurrences O1
LEFT JOIN Occurrences O2 ON O2.ObjectRef.Id_Object =
O1.ObjectRef.Id_Object AND O2.RegionRef.Id_Region =
O1.RegionRef.Id_Region AND O2.EventTypeRef.Id_Event = 1020
AND O1.Time_O < O2.Time_O
WHERE O1.EventTypeRef.Id_Event = 1010 AND
O1.ObjectRef.Id_Object = O AND O1.RegionRef.Id_Object = R
AND (SELECT COUNT(*) FROM Occurrences O3
WHERE O3.ObjectRef.Id_Object = O1.ObjectRef.Id_Object AND
O3.RegionRef.Id_Region = O1.RegionRef.Id_Region AND
O3.Time_O > O1.Time_O AND O3.Time_O < NVL(O2.Time_O,
NOW)=0
ORDER BY O1.Tiempo

```

Figure 16:9 SQL code to perform query in the question 3

7. CONCLUSIONS AND FUTURE WORKS

This paper presented an implementation for the snapshots + events spatiotemporal modelling approach with a case study to evaluate number of classical spatiotemporal queries. Traditional queries are answered with snapshots + events approach. This situation is not common to do with other approaches.

Snapshots + events approach is more efficient in practice than the traditional events modelling approach, for implementing and for answering some traditional queries, because some deductive characteristics are not so necessary. Although the evaluation was performed on a hypothetical case study, it was possible to demonstrate improved spatiotemporal query performance.

Further work on more realistic scenarios such as environmental monitoring will provide a real world practical application for evaluating the snapshots + events modelling approach.

8. ACKNOWLEDGEMENTS

We would like to acknowledge Prof. Philip Sallis, director of Geoinformatics Research Centre at AUT for his involvement, encouragements and support.

9. REFERENCES

- [1] Instituto Geográfico Militar. Atlas Geográfico de Chile para la Educación – Nueva Edición. Instituto Geográfico Militar, 2008.
- [2] M. P Armstrong. Temporality in spatial databases. Proceedings: GIS/LIS'88, 2:880-889, 1988.
- [3] H. Gregersen and C. S. Jensen. Temporal Entity-Relationship Models—A Survey. IEEE Transactions on Knowledge and Data Engineering, 11(3):464–497, 1999.
- [4] F.N. Kesim and Marek Sergot. A logic programming framework for modeling temporal objects. IEEE Transactions on Knowledge and Data Engineering, 8(5):724-741, 1996.
- [5] C. Vidal. Un Marco Formal para Base de Datos Espacio-Temporal basado en Objetos y Eventos. Master in Computer Sciences Theses, University of Concepción, 2007.
- [6] G. Gutierrez, G. Navarro, A. Rodríguez, A.González. A Spatio-Temporal Access Method based on Snapshots and Events. GIS'05, 2005, Bremen, Germany, November 4.
- [7] Oracle Technology Network. Oracle Database 10g Express Edition. <http://www.oracle.com/technology/products/database/xe/index.html>, 2010.
- [8] MySQL. MySQL :: The world's most popular open source database. <http://www.mysql.com>, 2010.
- [9] PostgreSQL. PostgreSQL: The world's most advanced open source database. <http://www.postgresql.org>, 2010.
- [10] Microsoft Express Database. SQL Server 2008 Express. <http://www.microsoft.com/express/database/>, 2010.
- [11] P. Grenin and B. Smith. Snap and span: Towards dynamic spatial ontology. Journal of Spatial Cognition and Computation, 4(1):69-103, 2004.
- [12] P. Grenon. The formal ontology of spatio-temporal reality and its formalization. AAAI, 2002.
- [13] Frederick E. Petry Thomas M. Schmidh and Roy Ladner. The object event calculus and temporal geographic information systems. 2003.
- [14] C. Vidal and M. A. Rodríguez. A Logical Approach for Modeling Spatio-Temporal Objects and Events. CoMoGIS'05, LNCS. 2005.
- [15] Paton, N.W., Fernandes, A.A.A. and Griffiths, T., Spatio-Temporal Databases: Contentions, Components and Consolidation, Int. Workshop on Advanced Spatial Databases (ASDM), 11th DEXA Workshop, A.M. Tjoa et al. (eds), IEEE Press, 851-855, 2000.