WORKING WITH SPATIO-TEMPORAL DATA TYPE

Ale Raza

Esri, 380 New York Street, Redlands, California 92373-8100 USA - araza@esri.com

Commission II, WG II/1

KEY WORDS: Databases, Standards, GIS, Object, Spatial, Temporal

ABSTRACT:

Several aspects of spatiotemporal databases have been explored in past decades, ranging from basic data structure to query processing and indexing. But today, operational temporal GIS does not exist. The key impediments have been the complexity of integrating space and time and the lack of standards. OpenGIS standards for simple feature access (spatial type) do exist, but unlike the spatial type, standards for spatiotemporal data type do not exist. This paper explores a new approach to modeling space and time to provide the basis for implementing a temporal GIS. This approach is based on the concept of data types in databases. A data type provides constructors, accessors, and operators. Most commercial and open source databases provide data types to deal with the spatial component of a GIS, called spatial type. Oracle Spatial, DB2 Spatial Extender and Informix Spatial DataBlade, ST_Geometry for PostgreSQL and Oracle from Esri, PostGIS for PostgreSQL, etc., are some examples. This new spatiotemporal data type is called spatiotemporal type (STT). This STT is implemented in PostgreSQL, an open source relational database management system. The STT is an extension of Esri's ST_Geometry type for PostgreSQL, in which each spatial object has a life span. Constructors, accessors, and relational functions are provided to create STT and support spatial, spatiotemporal, and temporal queries. Some functions are extended based on OpenGIS standards for the spatial type. Examples are provided to demonstrate the application of these functions. The paper concludes with limitations and challenges of implementing STT.

1. INTRODUCTION

Discussion on spatiotemporal databases is not new. Several aspects of spatiotemporal databases have been explored in the last two decades. There is a bewildering array of research ranging from simple/complex data structure to query language and spatiotemporal indexes. The space-time cube by Langran (1992), object-oriented approach to handle spatiotemporal objects by Worboys (1992), and cell tuple-based spatiotemporal data model by Raza and Kainz (1999) are major examples of data models or structures. Spatiotemporal query language by Snodgrass (1993) and spatiotemporal indexing techniques by Jensen et al. (2006) and Hadjieleftheriou et al., (2006) are other research examples. Detailed discussion on these researches is beyond the scope of this paper.

In spite of these developments, we do not see any operational temporal GIS. The key impediments have been attributed to the complexity of integrating space and time and the lack of standards for spatiotemporal data. Software standards ensure desirable characteristics of software. It ensures that it is implemented uniformly across the platforms and software inputs/outputs are known. OpenGIS standards for simple feature access (spatial type) do exist (OpenGIS, 2010) but unlike spatial type, standards for spatiotemporal data type do not exist. Many commercial and open source software conform to these standards.

This paper presents a new approach to modeling space and time to provide the basis for implementing a temporal GIS. This new approach is presented in section 2. Section 3 discusses various constructors, accessors, and operators for this new data type. Examples are provided in section 4. Limitations and further work are discussed in Section 5.

2. SPATIOTEMPORAL DATA TYPE

GIS is becoming an integral part of any enterprise database and no longer considered a separate branch from mainstream information technology (IT). As part of a standard database management system, a GIS is supposed to have a data type like any other data type in any database management system. For example, integer, double, or char are simple or basic data types. Database management systems also extend these basic data types to provide abstract data types (ADT) to handle more complex cases such as videos, CAD data, or spatial data. An ADT is a mathematical model to define data type. A data type is a collection of values and a set of operations on those values (Aaron et al., 1990). Recently, we have seen increasing support for spatial data types from commercial and open source database providers. Oracle Spatial from Oracle (Oracle11*g*, 2011), DB2 Spatial Extender (DB2, 2002) and Informix Spatial DataBlade from IBM (Informix, 2002), ST_Geometry for PostgreSQL and Oracle from Esri (ArcGIS 10, 2010), PostGIS for PostgreSQL from Refractions Research (PostGIS, 2010), SQL Server Geometry from Microsoft (SQL Server, 2008), and so forth, are some examples. Most of these implementations are OpenGIS simple feature access and SQL/MM specification compliant. While there are standards and implementations for spatial data types, we do not see the same for spatiotemporal data types.

This paper introduces a new data type for space-time modeling. A spatiotemporal data type is a data type where space and time is integrated and time is not considered an attribute of space. A similar approach was presented by Erwig et al. (1999) for moving objects. Basic objects in that approach were evolving points and regions. Line objects are considered as projections of movement. This approach is good for moving objects such as cars or airplanes but may not be best for urban applications such as land-use change or tracking parcel history where the object does not move rapidly. Moreover, it is not clear how to model line features such as roads in that approach.

The data model for STT is defined by an object model. In an object model, everything is an object. A class is a set of objects with common properties. A class consists of data members and member functions (operations). The structure of the data model is defined by data members (data), while operators and consistency rules are defined through operations.

Data structure and consistency rules for the model are discussed in this paper. The basic element of this model is a spatiotemporal object. This spatiotemporal class is the aggregation of spatial and temporal classes. The spatial and temporal object and associated operations are well understood in the respective domains and are not discussed here. This paper focuses on integration of spatial and temporal class.

Figure 1 depicts two classes: spatial and temporal. The object of a spatial class is a spatial object in Euclidean space while a temporal class represents the linear time.

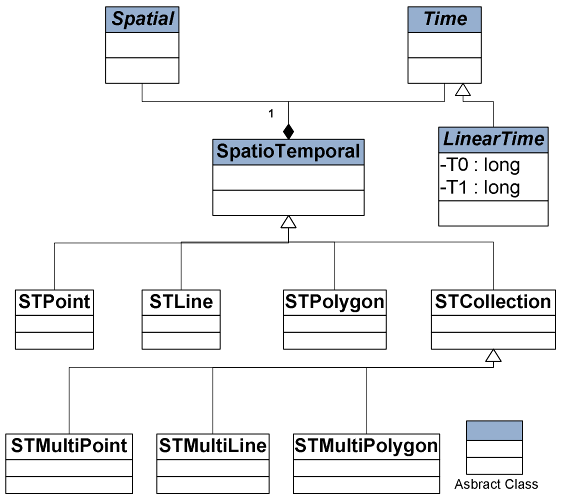


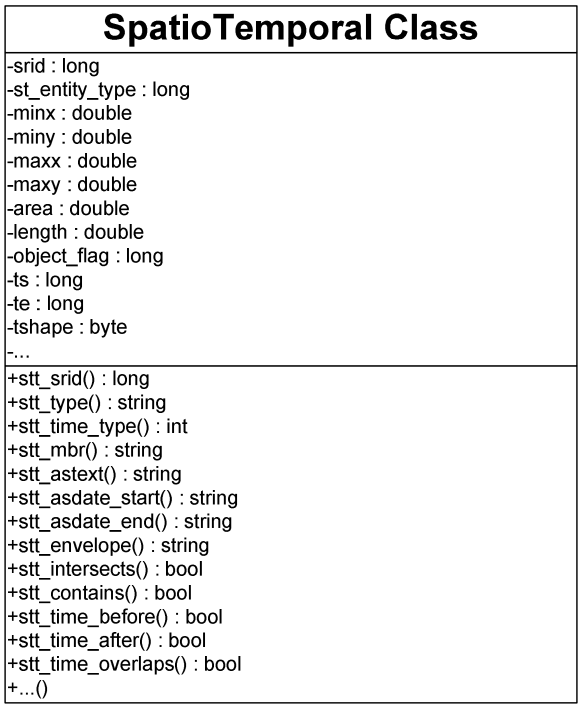
Figure 1. Spatialtemporal class

Only valid time called WorldTime (WT) is considered in this implementation. Figure 1 is the simplified ST data model. Each spatiotemporal object, STPolygon, STMultiPolygon, STLines, STMultiLines, STPoint, and STMultiPoint, has start (T_{Start}) and end (T_{End}) times.

The attributes and operations of a spatiotemporal class (ST-object) are described in figure 2. The operations can further be classified into three main categories: constructors, accessors, and operators. These are discussed in the next section.

In figure 2, srid is spatial reference system ID; st_entity_type is an entity type such as STPoint or STPolygon; minx, miny, maxx, and maxy are minimum and maximum coordinates of the spatiotemporal object; ts and te are start and end times, respectively; and tshape is the binary internal representation of the spatial part of a spatiotemporal object.

The schema for ST-type is presented in figure 3. The Feature schema has feature column type_column and other properties. The ST_Type schema contains information about the Feature schema and srid. The Spatial_References and Coordinate_Systems schemata store the information about srid.



SpatioTemporal Class

-srid : long
-st_entity_type : long
-minx : double
-miny : double
-maxx : double
-maxy : double
-area : double
-length : double
-object_flag : long
-ts : long
-te : long
-tshape : byte
-...

+stt_srid() : long
+stt_type() : string
+stt_time_type() : int
+stt_mbr() : string
+stt_astext() : string
+stt_asdate_start() : string
+stt_asdate_end() : string
+stt_envelope() : string
+stt_intersects() : bool
+stt_contains() : bool
+stt_time_before() : bool
+stt_time_after() : bool
+stt_time_overlaps() : bool
+...()

Figure 2. Spatiotemporal class attributes and operations

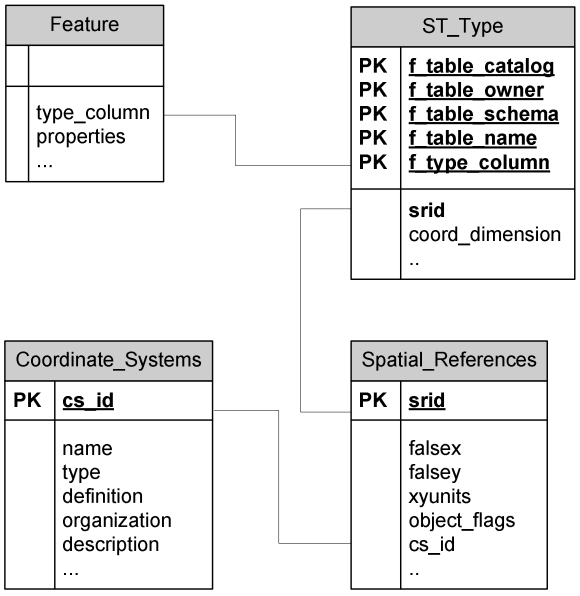


Figure 3. ST-type schema

3. CONSTRUCTORS, ACCESSORS, AND OPERATORS

As mentioned earlier, a type consists of constructor, accessor, and operator functions. A constructor is used to create a type. Two types of constructors are provided:

- STT_Type(WKTwT, SRID)
- STT_Type(WKB, SRID, T)

Where WKTwT (Well Known Text with Time) is the extension of OpenGIS WKT (Well Known Text) format. WKTwT consists of two parts: WKT and Time text.

where
WKTwT = WKT: T and
T time can be interval (T1) or instance (T0)
$T1 = \{T_{Start},T_{End} \mid (T_{Start} \leq T_{End}) \wedge (T_{Start}T_{End} \in WT)\}$
$T0 = \{T_{Start},T_{End} \mid (T_{Start} = T_{End}) \wedge (T_{Start}T_{End} \in WT)\}$

T is represented in ISO 8601format (YYYY-MM-DD) as recommended by PostgreSQL (PostgreSQL, 2011).
Therefore, WKTwT for point data can be represented as

1. WKTwT = 'POINT (10 20): 1970-01-01 1970-01-01'
2. WKTwT = 'POINT (10 20): 1970-01-01 '
3. WKTwT = 'POINT (10 20): 1970-01-01 1975-12-31'
4. WKTwT = 'POINT (10 20): 1970-01-01 NOW'

The first two formats (#1 and #2) represent time instance (T0), while the rest of the format shows the time interval (T1). 'NOW' represents the open time, which means the object is still alive. At this time, only the date format is supported, time is not included. The binary format can take WKB (Well Known Binary). Other formats for constructors are

- STT_TypeFromShape(Esri_shape_format, SRID, T)
- STT_TypeFromWKB(WKB, SRID, T)

Shape data can be returned in various formats by using different type output accessor's functions. Some of these functions are

- STT_AsText(st_type)
- STT_AsBinary(st_type)
- STT_AsShape(st_type)

Only the first STT_AsText function returns spatial data with time in WKTwT format. Whereas STT_AsBinary and STT_AsShape return results in WKB and Esri shape format, respectively. Similarly, accessor's functions can be used to extract type information. Here are some examples:

- STT_Envelope
- STT_AsDate_Start
- STT_AsDate_End

For spatiotemporal type, operators can be divided into three categories: pure temporal (table 1), pure spatial, and spatiotemporal operators (table 2).

Spatial and temporal are regular operators employed in spatial and temporal databases. Temporal relations are analogs to spatial topological relations but a bit different than spatial topological relations. The former has the sense of direction, or in other words, these relations are ordered topological relations.

Table 1 illustrates temporal relations. STTF_time_overlaps is a special case where oredering is ignored. Pure spatial and temporal relations start with the prefix STTF_, while spatiotemporal relations start with the prefix STT_.

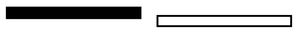
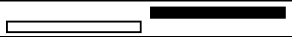
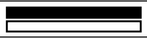
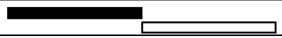
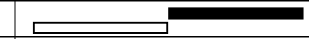
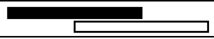
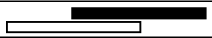
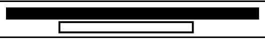
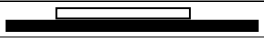
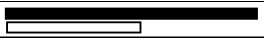
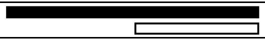
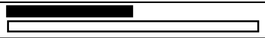
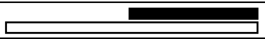
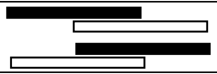
	Temporal Relations	Illustration t1 = ▬ t2 = ▭
1	STTF_time_before	
2	STTF_time_after	
3	STTF_time_equal	
4	STTF_time_meets	
5	STTF_time_met	
6	STTF_time_left_overlaps	
7	STTF_time_right_overlaps	
8	STTF_time_covers	
9	STTF_time_covered	
10	STTF_time_right_covers	
11	STTF_time_left_covers	
12	STTF_time_right_covered	
13	STTF_time_left_covered	
14	STTF_time_overlaps	

Table 1. Adapted temporal relations for bounded interval (Raza, 2008)

Let t1 and t2 are two time intervals and $\{\varnothing\}$ is an empty set. Mathematically, relations in table 1 can be formulized as

1. t1.STTF_time_before(t2) ⇔
$\{(t1 \cap t2 = \varnothing) \mid (t1_e < t2_s) \wedge (t1, t2 \in WT)\}$

2. t1.STTF_time_after(t2) ⇔
$\{(t1 \cap t2 = \varnothing) \mid (t1_s > t2_e) \wedge (t1, t2 \in WT)\}$

3. t1.STTF_time_equal(t2) ⇔
$\{(t1 \cap t2 \neq \varnothing) \mid (t1_s = t2_s) \wedge (t1_e = t2_e) \wedge (t1, t2 \in WT)\}$

4. t1.STTF_time_meets(t2) ⇔
$\{(t1 \cap t2 \neq \varnothing) \mid (t1_e = t2_s) \wedge (t1, t2 \in WT)\}$

5. t1.STTF_time_met(t2) ⇔
$\{(t1 \cap t2 \neq \varnothing) \mid (t1_s = t2_e) \wedge (t1_e > t2_e) \wedge (t1, t2 \in WT)\}$

6. t1.STTF_time_left_overlaps(t2) ⇔
$\{(t1 \cap t2 \neq \varnothing) \mid (t1_e > t2_s) \wedge (t1_e < t2_e) \wedge (t1, t2 \in WT)\}$

7. t1.STTF_time_right_overlaps(t2) ⇔
$\{(t1 \cap t2 \neq \varnothing) \mid (t1_s > t2_s) \wedge (t1_e > t2_e) \wedge (t1, t2 \in WT)\}$

8. t1.STTF_time_covers(t2) ⇔
$\{(t1 \cap t2 \neq \varnothing) \mid (t1_s < t2_s) \wedge (t1_e > t2_e) \wedge (t1, t2 \in WT)\}$

9. t1.STTF_time_covered(t2) ⇔
$\{(t1 \cap t2 \neq \varnothing) \mid (t1_s > t2_s) \wedge (t1_e < t2_e) \wedge (t1, t2 \in WT)\}$

10. t1.STTF_time_right_covers(t2) ⇔
$\{(t1 \cap t2 \neq \varnothing) \mid (t1_s = t2_s) \wedge (t1_e > t2_e) \wedge (t1, t2 \in WT)\}$

11. t1.STTF_time_left_covers(t2) ⇔
$\{(t1 \cap t2 \neq \varnothing) \mid (t1_e = t2_s) \wedge (t1_e < t2_e) \wedge (t1, t2 \in WT)\}$

12. t1.STTF_time_right_covered(t2) ⇔
$\{(t1 \cap t2 \neq \emptyset) \mid (t1_s = t2_s) \wedge (t1_e < t2_e) \wedge (t1, t2 \in WT)\}$

13. t1.STTF_time_left_covered(t2) ⇔
$\{(t1 \cap t2 \neq \emptyset) \mid (t1_s > t2_s) \wedge (t1_e = t2_e) \wedge (t1, t2 \in WT)\}$

14. t1.STTF_time_overlaps(t2) ⇔
$\{(t1 \cap t2 \neq \emptyset) \mid (t1, t2 \in WT)\}$

In case of instance time T0, these relations collapse to only three, in other words, STTF_time_before, STTF_time_after, and STTF_time_overlap. In this paper, spatiotemporal operators are introduced to discern spatial relation over time. Table 2 lists some of these operators.

Spatiotemporal Operators	
STT_contains	STT_time_left_covers
STT_within	STT_time_covers
STT_intersects	STT_time_right_covered
STT_overlaps	STT_time_equal
STT_touches	STT_time_right_covers
STT_crosses	STT_time_covered
STT_equals	STT_time_left_covered
STT_disjoint	STT_time_right_overlaps
STT_time_before	STT_time_met
STT_time_meets	STT_time_after
STT_time_left_overlaps	STT_time_overlaps

Table 2. Spatiotemporal operators

Operators in table 2 can formally be defined as follows: let *A* and *B* are two st_type objects, *g* is geometry, *t* is time, α is boundaries, β is interior, and γ is exterior of st_type object. Then, the operation on st_type can be defined as

- *A*.STT_intersects(*B*) ⇔
$((A.g \cap B.g \neq \emptyset) \wedge (A.t \cap B.t \neq \emptyset))$

- *A*.STT_ overlaps (*B*) ⇔
$((\beta(A) \cap \beta(B) \neq \emptyset) \wedge (\beta(A) \cap \gamma(B) \neq \emptyset) \wedge$
$(\gamma(A) \cap \beta(B) \neq \emptyset) \wedge (A.t \cap B.t \neq \emptyset))$

Where pure spatial relation as

- *A*.STTF_intersects(*B*) ⇔
$((A.g \cap B.g \neq \emptyset))$

- *A*.STTF_ overlaps (*B*) ⇔
$((\beta(A) \cap \beta(B) \neq \emptyset) \wedge (\beta(A) \cap \gamma(B) \neq \emptyset) \wedge$
$(\gamma(A) \cap \beta(B) \neq \emptyset))$

The spatiotemporal STT_time_overlaps can be defined as

- t1.STT_time_overlaps(t2) ⇔
$\{(t1 \cap t2 \neq \emptyset) \wedge ((\beta(A) \cap \beta(B) \neq \emptyset) \wedge (\beta(A) \cap \gamma(B) \neq \emptyset) \wedge$
$(\gamma(A) \cap \beta(B) \neq \emptyset) \wedge (A.t \cap B.t \neq \emptyset)) \mid (t1, t2 \in WT)\}$

Describing all functions is beyond the scope of this paper. These above functions are shown to demonstrate the difference between spatial and spatiotemporal operators.

The spatiotemporal data type is implemented in PostgreSQL 9.1.2 (PostgreSQL, 2011). The STT is an extension of Esri's ST_Geometry type for PostgreSQL. Section 4 describes some implementation examples.

4. SOME EXAMPLES

The parcel data of San Diego, California, is used to show how queries can be performed to fetch time series data. The actual data was not time series data. There are about 800,000 records in this database. These parcels are distributed as shown in table 3.

T_{Start}	T_{End}	Number of parcels
1970-01-01	1972-12-31	500
1973-01-01	1974-12-31	500
1975-01-01	1977-12-31	500
1978-01-01	1980-12-31	500
1981-01-01	1983-12-31	500
1984-01-01	1986-12-31	500
1987-01-01	1989-12-31	97000
1990-01-01	1991-12-31	100000
1992-01-01	1993-12-31	100000
1994-01-01	1995-12-31	100000
1996-01-01	1997-12-31	100000
1998-01-01	1999-12-31	100000
2000-01-01	2001-12-31	100000
2002-01-01	NOW	83159
Total		783159

Table 3. Number of parcels by date

To demonstrate the concept, let's create a simple parcel table to insert and query data.

- Create table parcel (objectid serial, shape st_type);

Constructors can be used to insert st_type into the database. For example, the ST_Type constructors can be used to insert parcel data with time into the parcel table.

```
Insert Into Parcel (shape) Values
(ST_Type('POLYGON                          ⎫
 ((6150763 1775304, 6150763 2129759,        ⎬ geometry
 6613437 2129759, 6613437 1775304,          ⎪
 6150763 1775304)):                         ⎭
 1970-01-01 1971-12-31',                   ➔ time
 4326);                                    ➔ srid
```

Similarly, queries can be performed using accessors and operators using various filters.

- Time filters
    - Simple filter
    - Range filter
- Spatial filters
- Spatiotemporal filters
    - Simple filter
    - Range filter

Range queries are window-based queries. This window can define the spatial or temporal extent. Here spatiotemporal queries are discussed.

1. Simple spatiotemporal filters

Query-1: Find all parcels in the year 1980.

```
Query: Select Count(*) From Parcel
 Where STT_Time_Covers(shape,
 ST_Type('POLYGON
 ((6150763 1775304, 6150763 2129759,
 6613437 2129759, 6613437 1775304,
 6150763 1775304)):
 1980-01-01 ',
 4326)) = 'true';
```

Result: 500

In this case, the polygon is the maximum extent of all geometries.

Query 2: Show all parcels in the year 1970 that intersect with the given window. In this case, the window is represented by a polygon.

```
Query: Select STT_AsText(Shape) from Parcel
 Where STT_Intersects(shape,
 ST_Type('POLYGON
 ((6300666 2076508, 6301123 2076508,
 6301123 2077952, 6300666 2077952,
 6300666 2076508)):
 1970-01-01 ',4326)) = 'true';
```

Result: POLYGON ((6300666 2077931, ..., 6301123 2077952, 6300666 2077931)): 1970-01-01 1972-12-31

2. Range Spatiotemporal filters

Query 3: Find all parcels during '1990-01-01 1991-12-31' date.

```
Query: Select Count(*) From Parcel
 Where STT_Time_Overlaps(shape,
 ST_Type('POLYGON
 ((6150763 1775304, 6150763 2129759,
 6613437 2129759, 6613437 1775304,
 6150763 1775304)):
 1990-01-01 1991-12-31',
 4326)) = 'true';
```

Result: 100000

Query 4: Show all parcels during '2000-01-01 2001-12-31' date that intersect with given envelope.

```
Query: Select ST_AsText(Shape) From Parcel
 Where STT_Intersects(shape,
 ST_Type('POLYGON
 ((6288821 1856272, 6288962 1856272,
 6288962 1856326, 6288821 1856326,
 6288821 1856272)):
 2000-01-01 2001-12-31',
 4326)) = 'true';
```

Result: POLYGON ((6288821 1856325, 6288821 1856273, 6288961 1856272, 6288962 1856325, 6288821 1856325)): 2000-01-01 2001-12-31

Queries 1 and 3 are temporal range queries where the spatial range is set to the maximum spatial extent of all geometries. Similarly, temporal range can be set to the maximum temporal extent to work with spatial range queries. For example,

Query 5: Find all parcels within given spatial extent.

```
Query: Select Count(*) From Parcel
 Where STT_Intersects(shape,
 ST_Type('POLYGON
 ((6314002 1859345, 6354405 1859345,
 6354405 1882153, 6314002 1882153,
 6314002 1859345)):
 1969-01-01 NOW',
 4326)) = 'true';
```

Result: 40648

Query 7: How many parcels were created before 1975?

```
Query: Select Count(*) From Parcel
 Where STT_Time_Before(shape,
 ST_Type('POLYGON
 ((6314002 1859345, 6354405 1859345,
 6354405 1882153, 6314002 1882153,
 6314002 1859345)):
 1975-01-01 ',
 4326)) = 'true' ;
```

Result: 1000

Query 6: How many parcels are not closed yet?

```
Query: Select Count(*) From Parcel
 Where STT_Time_left_Covers(shape,
 ST_Type('POLYGON
 ((6314002 1859345, 6354405 1859345,
 6354405 1882153, 6314002 1882153,
 6314002 1859345)):
 1969 NOW',
 4326)) = 'true';
```

Result: 83159

These are some of the simple examples of operator and accessor functions. Of course, more complex queries can be formed to answer more complex questions.

5. CONCLUSION

The paper presents some initial results of ongoing research in spatiotemporal data type. This research is still in its early stage. A new data type is presented in this paper to deal with complex issues of space-time modeling. The list of operator, constructor, and accessor functions is not complete and more work is needed to document remaining functions. Currently, binary data is returned in WKB or Esri shape format (data without time). There is a need to return binary data in a format where time is also stored. Future work involves indexing this data type. Without indexing, performance cannot be achieved. Full implementation of this data type would help us implement temporal GIS.

References

Tenenbaum, Aaron M., Yedidyah Langsam, and Moshe J. Augenstein, 1990. *Data structures using C*. Prentice-Hall, New Jersey, pp. 13–14.

ArcGIS 10, 2010, ArcGIS Help Library, http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#//006z00000050000000, (December 27, 2011).

DB2, 2002. IBM DB2 Spatial Extender, User's Guide and Reference, Version 8, ftp://public.dhe.ibm.com/ps/products/db2/info/vr8/pdf/letter/db2sbe80.pdf (December 27, 2011).

Erwig, M., R. Güting, M. Schneider, and M. Vazirgiannis, 1999. Spatio-Temporal Data Types: An Approach to Modeling and Querying Moving Objects in Databases, *GeoInformatica,* Vol. 3, No. 3, pp. 269–296.

Hadjieleftheriou, M., G. Kollios, V. Tsotras, and D. Gunopulos, 2006. Indexing Spatio-temporal Archives, *The International Journal on Very Large Data Bases*, Volume 15, Issue 2, pp. 143 - 164.

Informix, 2002. IBM Informix Spatial DataBlade Module User's Guide, Version 8.20, http://publib.boulder.ibm.com /epubs/pdf/9119.pdf (December 27, 2011).

Jensen, C., D. Tiesyte, and N. Tradisauskas, 2006. The COST Benchmark-Comparison and Evaluation of Spatio-temporal Indexes, In *Database Systems for Advanced Applications*. Springer Berlin, pp. 125–140.

Langran, G., 1992, *Time in geographic information systems*, Taylor and Francis Ltd.

Oracle11*g*, 2011. Oracle Database Concepts, http://docs.oracle.com/cd/B28359_01/server.111/b28318/conte nt.htm#BAJJIFJJ (January 11, 2012).

OpenGIS, 2010. OpenGIS Implementation Standard for Geographic information—Simple feature access—Part 2: SQL option, http://www.opengeospatial.org/standards/sfs (December 27, 2011).

PostGIS, 2010. PostGIS Manual, Version 1.5.2, http://postgis.refractions.net/download/postgis-1.5.3.pdf (December 27, 2011).

PostgreSQL, 2011. PostgreSQL 9.1.2 Documentation. http://www.postgresql.org/docs/9.1/static/datatype-datetime.html (December 27, 2011).

Raza, A., 2008. Driving spatiotemporal relations from simple data structure, *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Vol. XXXVII. Part B2. Beijing 2008*, pp. 13-18

Raza, A., and W. Kainz, 1999. Cell tuple based spatio-temporal data model: An object oriented approach. In: *The Proceedings of the Eighth ACM Conference on Information and Knowledge Management (CIKM'99)* and *Symposium on Geographic Information Systems (GIS'99)*, Kansas City, Missouri, USA, pp. 20–25.

Snodgrass, R., 1993, An overview of TQuel, In *Temporal databases: theory, design and implementation*, Edited by A. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. Snodgrass, Benjamin/Cummings Publishing Company, pp. 141–182.

SQL Server, 2008. SQL Server 2008 R2, MSDN Library, http://msdn.microsoft.com/en-us/library/bb964711.aspx (December 27, 2011).

Worboys, M., 1992. Object-oriented models of spatiotemporal information. In: *The Proceedings of the GIS/LIS,* San Jose, CA, USA, pp. 825–834.